Making Health Economics Shiny: A tutorial

Robert Smith & Paul Schneider

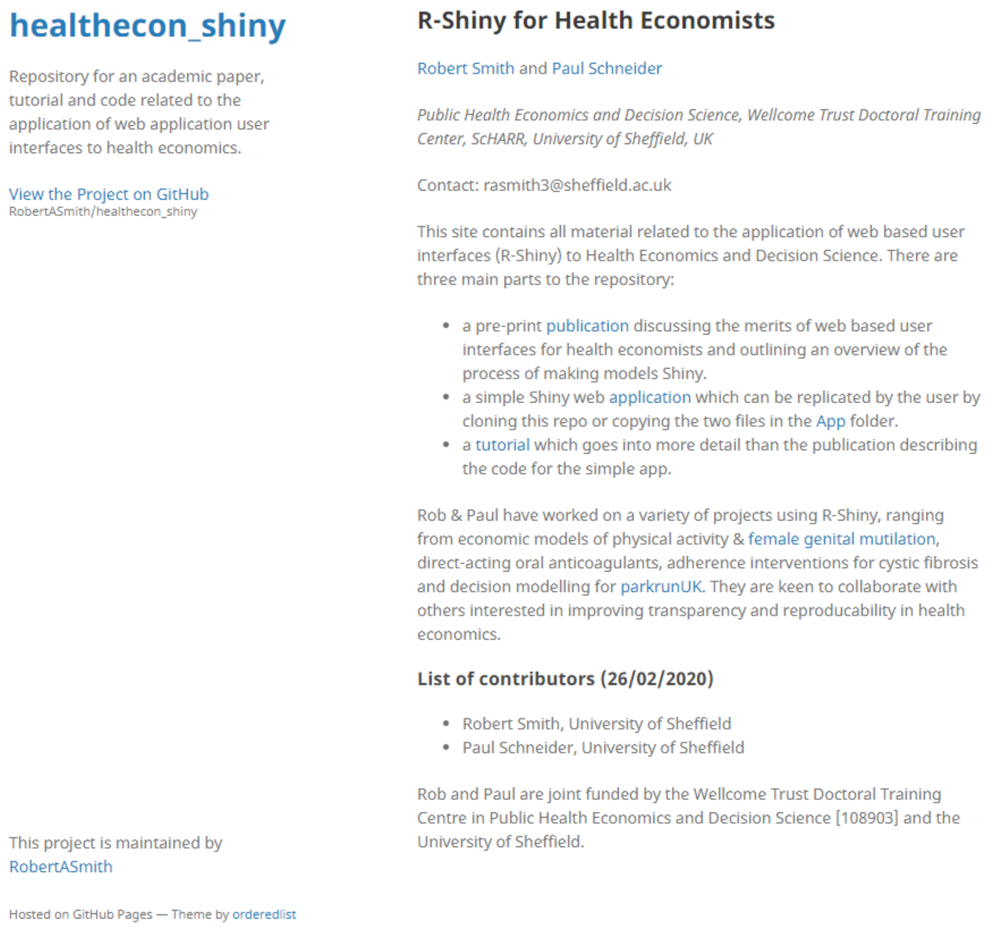
ScHARR, University of Sheffield

Tutorial Information

Open Access Content Available here:
https://robertasmith.github.io/healthecon_shiny/

Including:

- Academic Paper
- Tutorial
- App code



healthecon_shiny

Repository for an academic paper, tutorial and code related to the application of web application user interfaces to health economics.

View the Project on GitHub
RobertASmith/healthecon_shiny

R-Shiny for Health Economists

Robert Smith and Paul Schneider

Public Health Economics and Decision Science, Wellcome Trust Doctoral Training Center, ScHARR, University of Sheffield, UK

Contact: rasmith3@sheffield.ac.uk

This site contains all material related to the application of web based user interfaces (R-Shiny) to Health Economics and Decision Science. There are three main parts to the repository:

- a pre-print publication discussing the merits of web based user interfaces for health economists and outlining an overview of the process of making models Shiny.
- a simple Shiny web application which can be replicated by the user by cloning this repo or copying the two files in the App folder.
- a tutorial which goes into more detail than the publication describing the code for the simple app.

Rob & Paul have worked on a variety of projects using R-Shiny, ranging from economic models of physical activity & female genital mutilation, direct-acting oral anticoagulants, adherence interventions for cystic fibrosis and decision modelling for parkrunUK. They are keen to collaborate with others interested in improving transparency and reproducibility in health economics.

List of contributors (26/02/2020)

- Robert Smith, University of Sheffield
- Paul Schneider, University of Sheffield

Rob and Paul are joint funded by the Wellcome Trust Doctoral Training Centre in Public Health Economics and Decision Science [108903] and the University of Sheffield.

This project is maintained by RobertASmith

Hosted on GitHub Pages — Theme by orderedlist

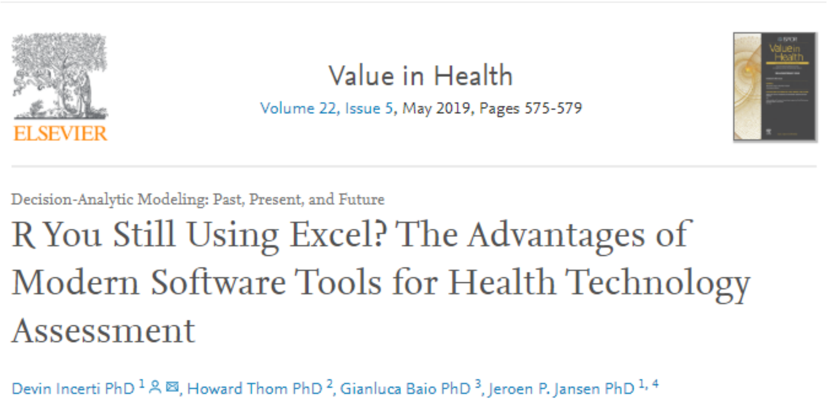
Prerequisite knowledge

- R programming:
  - Functions
  - Objects
  - Loops
  - Packages
  - Plotting

- Health economics:
  - Markov Model
  - QALYs
  - ICERs
  - PSA
  - Cost-effectiveness Plane
  - CEAC

Background

Background



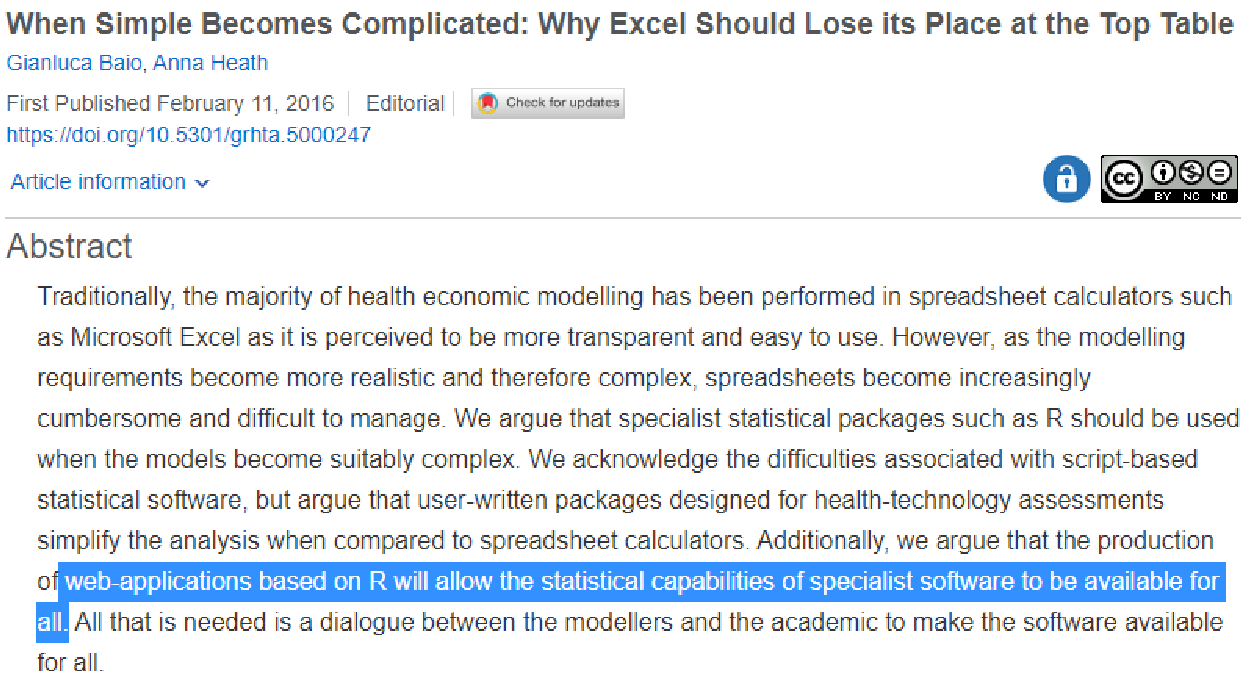
ELSEVIER

Value in Health
Volume 22, Issue 5, May 2019, Pages 575-579

Decision-Analytic Modeling: Past, Present, and Future

R You Still Using Excel? The Advantages of Modern Software Tools for Health Technology Assessment

Devin Incerti PhD [1], Howard Thom PhD [2], Gianluca Baio PhD [3], Jeroen P. Jansen PhD [1, 4]



When Simple Becomes Complicated: Why Excel Should Lose its Place at the Top Table

Gianluca Baio, Anna Heath

First Published February 11, 2016 | Editorial | Check for updates
https://doi.org/10.5301/grhta.5000247

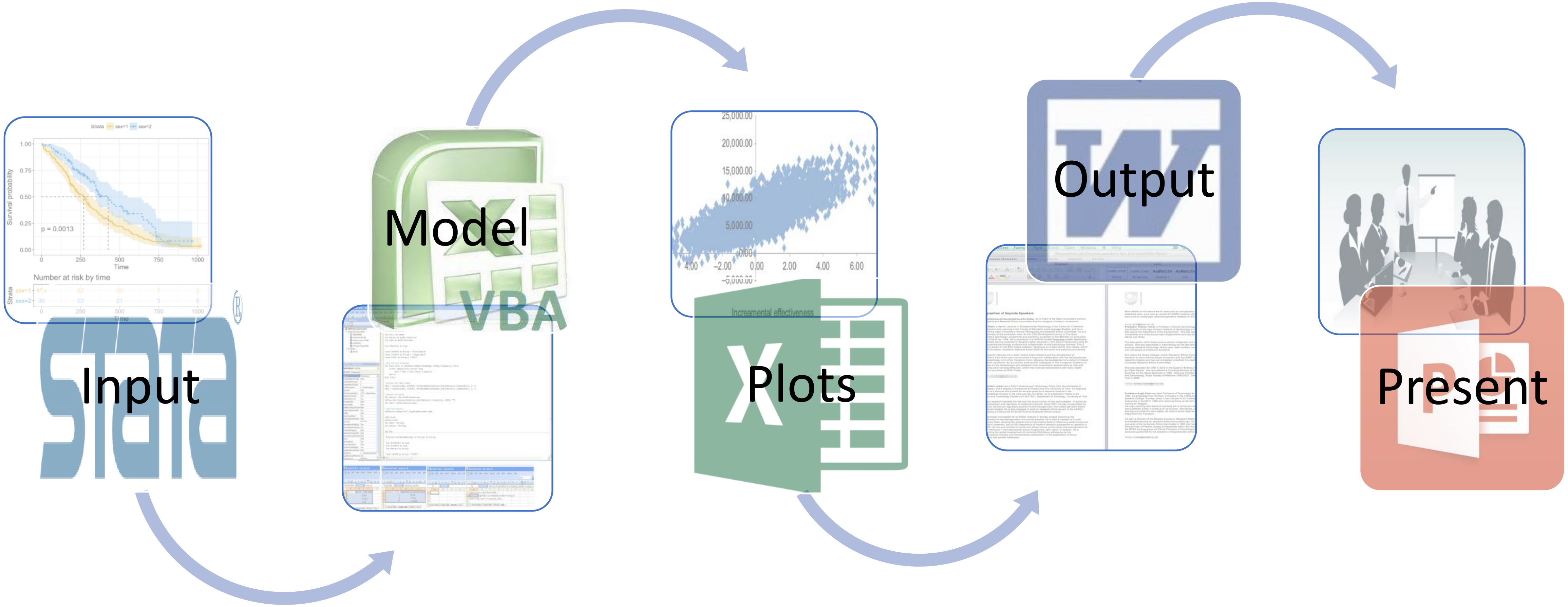
Article information ⌄

Abstract

Traditionally, the majority of health economic modelling has been performed in spreadsheet calculators such as Microsoft Excel as it is perceived to be more transparent and easy to use. However, as the modelling requirements become more realistic and therefore complex, spreadsheets become increasingly cumbersome and difficult to manage. We argue that specialist statistical packages such as R should be used when the models become suitably complex. We acknowledge the difficulties associated with script-based statistical software, but argue that user-written packages designed for health-technology assessments simplify the analysis when compared to spreadsheet calculators. Additionally, we argue that the production of web-applications based on R will allow the statistical capabilities of specialist software to be available for all. All that is needed is a dialogue between the modellers and the academic to make the software available for all.

“The future of cost-effectiveness modelling lies in web-apps, in which graphical interfaces are used to run script-based models”

Current Process



Input

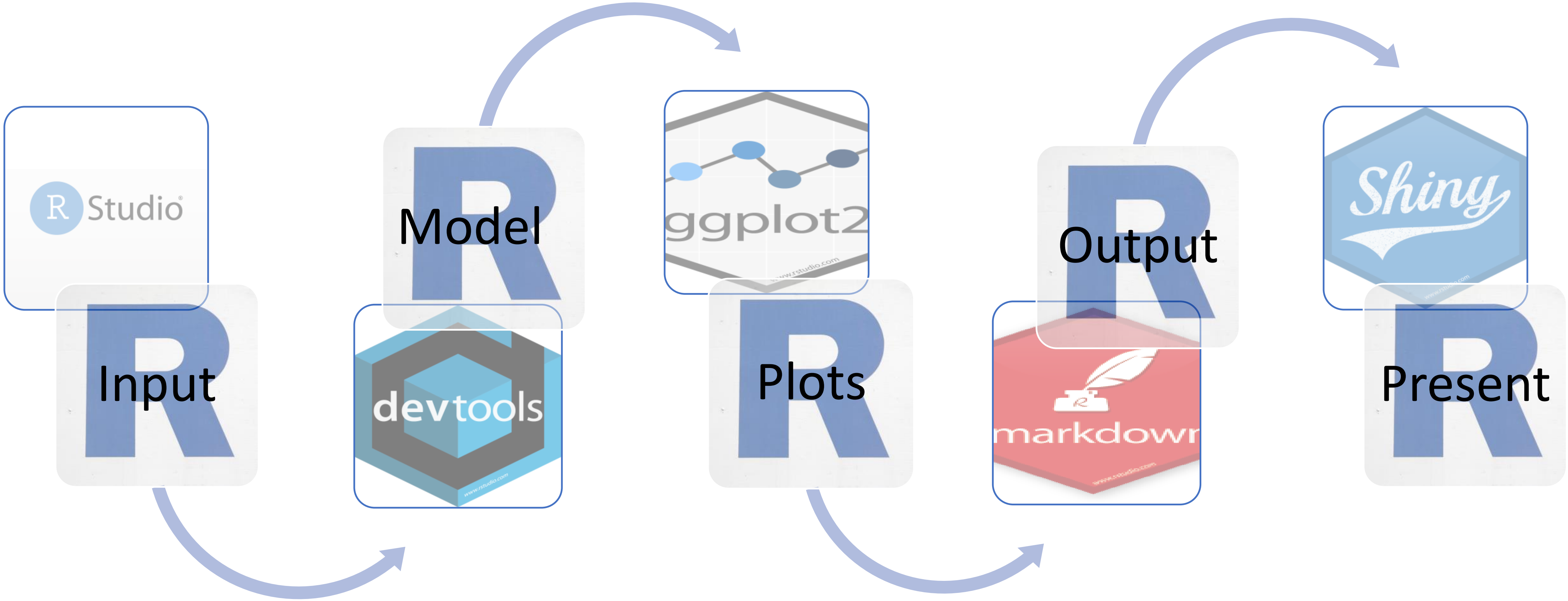
Model

Plots

Output

Present

Future process?



Input

Model

Plots

Output

Present

R for Decision Modelling?

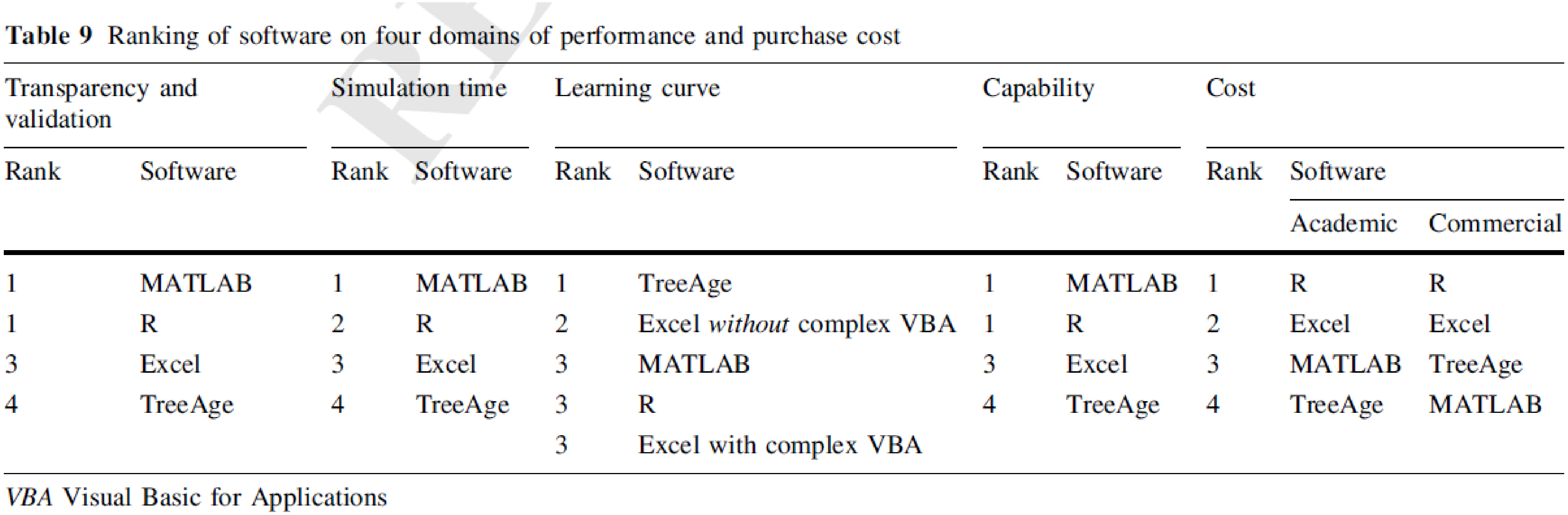


Table 9 Ranking of software on four domains of performance and purchase cost

Transparency and validation		Simulation time		Learning curve		Capability		Cost		
Rank	Software	Rank	Software	Rank	Software	Rank	Software	Rank	Software	
									Academic	Commercial
1	MATLAB	1	MATLAB	1	TreeAge	1	MATLAB	1	R	R
1	R	2	R	2	Excel *without* complex VBA	1	R	2	Excel	Excel
3	Excel	3	Excel	3	MATLAB	3	Excel	3	MATLAB	TreeAge
4	TreeAge	4	TreeAge	3	R	4	TreeAge	4	TreeAge	MATLAB
				3	Excel with complex VBA					

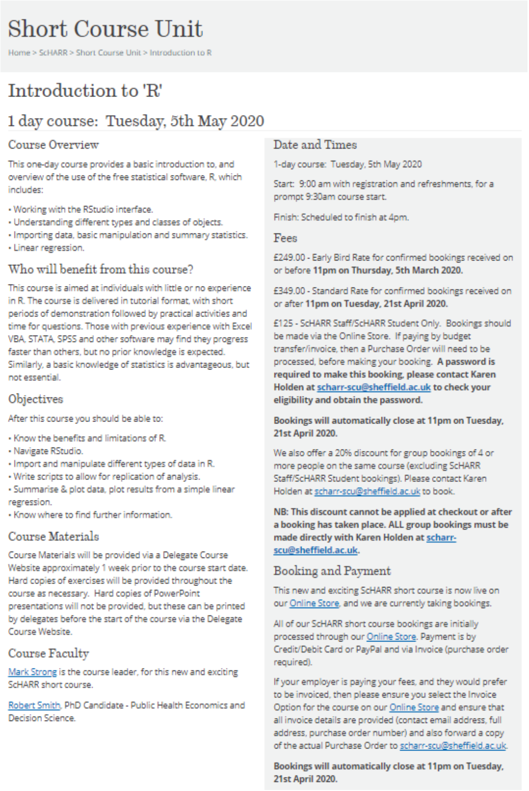
VBA Visual Basic for Applications

Hollman, C., Paulden, M., Pechlivanoglou, P. and McCabe, C., 2017. A comparison of four software programs for implementing decision analytic cost-effectiveness models. *Pharmacoeconomics*, *35*(8), pp.817-830.

Teaching and Tutorials

Decision Analysis in R for Technologies in Health

A collaborative for open-source, transparent decision analytic modelling


Short Course Unit

Introduction to 'R'

1 day course: Tuesday, 5th May 2020

Course Overview

This one-day course provides a basic introduction to, and overview of the use of the free statistical software, R, which includes:

- Working with the RStudio interface.
- Understanding different types and classes of objects.
- Importing data, basic manipulation and summary statistics.
- Linear regression.

Who will benefit from this course?

This course is aimed at individuals with little or no experience in R. The course is delivered in tutorial format, with short periods of demonstration followed by practical activities and time for questions. Those with previous experience with Excel VBA, STATA, SPSS and other software may find they progress faster than others, but no prior knowledge is expected. Similarly, a basic knowledge of statistics is advantageous, but not essential.

Objectives

After this course you should be able to:

- Know the benefits and limitations of R.
- Navigate RStudio.
- Import and manipulate different types of data in R.
- Write scripts to allow for replication of analysis.
- Summarise & plot data, plot results from a simple linear regression.
- Know where to find further information.

Course Materials

Course Materials will be provided via a Delegate Course Website approximately 1 week prior to the course start date. Hard copies of exercises will be provided throughout the course as necessary. Hard copies of PowerPoint presentations will not be provided, but these can be printed by delegates before the start of the course via the Delegate Course Website.

Course Faculty

Mark Strong is the course leader, for this new and exciting ScHARR short course.

Robert Smith, PhD Candidate - Public Health Economics and Decision Science.

Date and Times

1-day course: Tuesday, 5th May 2020

Start: 9:00 am with registration and refreshments, for a prompt 9:30am course start.

Finish: Scheduled to finish at 4pm.

Fees

£249.00 - Early Bird Rate for confirmed bookings received on or before **11pm on Thursday, 5th March 2020.**

£349.00 - Standard Rate for confirmed bookings received on or after **11pm on Tuesday, 21st April 2020.**

£125 - ScHARR Staff/ScHARR Student Only. Bookings should be made via the Online Store. If paying by budget transfer/invoice, then a Purchase Order will need to be processed, before making your booking. **A password is required to make this booking, please contact Karen Holden at scharr-scu@sheffield.ac.uk to check your eligibility and obtain the password.**

Bookings will automatically close at 11pm on Tuesday, 21st April 2020.

We also offer a 20% discount for group bookings of 4 or more people on the same course (excluding ScHARR Staff/ScHARR Student bookings). Please contact Karen Holden at scharr-scu@sheffield.ac.uk to book.

NB: This discount cannot be applied at checkout or after a booking has taken place. ALL group bookings must be made directly with Karen Holden at scharr-scu@sheffield.ac.uk.

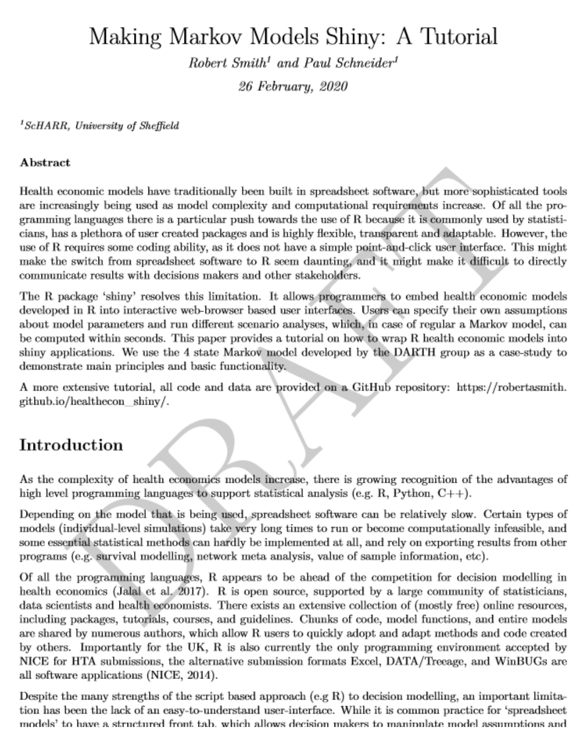
Booking and Payment

This new and exciting ScHARR short course is now live on our Online Store, and we are currently taking bookings.

All of our ScHARR short course bookings are initially processed through our Online Store. Payment is by Credit/Debit Card or PayPal and via Invoice (purchase order required).

If your employer is paying your fees, and they would prefer to be invoiced, then please ensure you select the Invoice Option for the course on our Online Store and ensure that all invoice details are provided (contact email address, full address, purchase order number) and also forward a copy of the actual Purchase Order to scharr-scu@sheffield.ac.uk.

Bookings will automatically close at 11pm on Tuesday, 21st April 2020.


Making Markov Models Shiny: A Tutorial

Robert Smith[1] and Paul Schneider[1]

26 February, 2020

[1] *ScHARR, University of Sheffield*

Abstract

Health economic models have traditionally been built in spreadsheet software, but more sophisticated tools are increasingly being used as model complexity and computational requirements increase. Of all the programming languages there is a particular push towards the use of R because it is commonly used by statisticians, has a plethora of user created packages and is highly flexible, transparent and adaptable. However, the use of R requires some coding ability, as it does not have a simple point-and-click user interface. This might make the switch from spreadsheet software to R seem daunting, and it might make it difficult to directly communicate results with decisions makers and other stakeholders.

The R package 'shiny' resolves this limitation. It allows programmers to embed health economic models developed in R into interactive web-browser based user interfaces. Users can specify their own assumptions about model parameters and run different scenario analyses, which, in case of regular a Markov model, can be computed within seconds. This paper provides a tutorial on how to wrap R health economic models into shiny applications. We use the 4 state Markov model developed by the DARTH group as a case-study to demonstrate main principles and basic functionality.

A more extensive tutorial, all code and data are provided on a GitHub repository: https://robertasmith.github.io/healthecon_shiny/.

Introduction

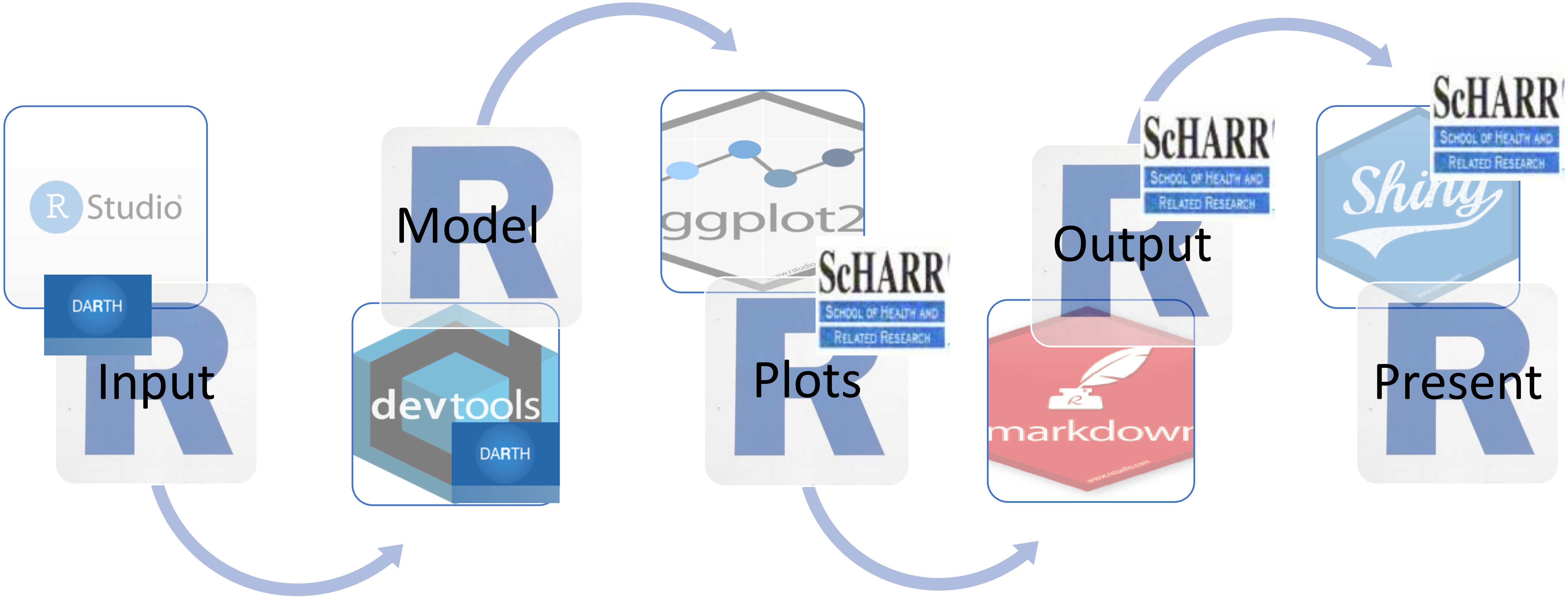
As the complexity of health economics models increase, there is growing recognition of the advantages of high level programming languages to support statistical analysis (e.g. R, Python, C++).

Depending on the model that is being used, spreadsheet software can be relatively slow. Certain types of models (individual-level simulations) take very long times to run or become computationally infeasible, and some essential statistical methods can hardly be implemented at all, and rely on exporting results from other programs (e.g. survival modelling, network meta analysis, value of sample information, etc).

Of all the programming languages, R appears to be ahead of the competition for decision modelling in health economics (Jalal et al. 2017). R is open source, supported by a large community of statisticians, data scientists and health economists. There exists an extensive collection of (mostly free) online resources, including packages, tutorials, courses, and guidelines. Chunks of code, model functions, and entire models are shared by numerous authors, which allow R users to quickly adopt and adapt methods and code created by others. Importantly for the UK, R is also currently the only programming environment accepted by NICE for HTA submissions, the alternative submission formats Excel, DATA/Treeage, and WinBUGs are all software applications (NICE, 2014).

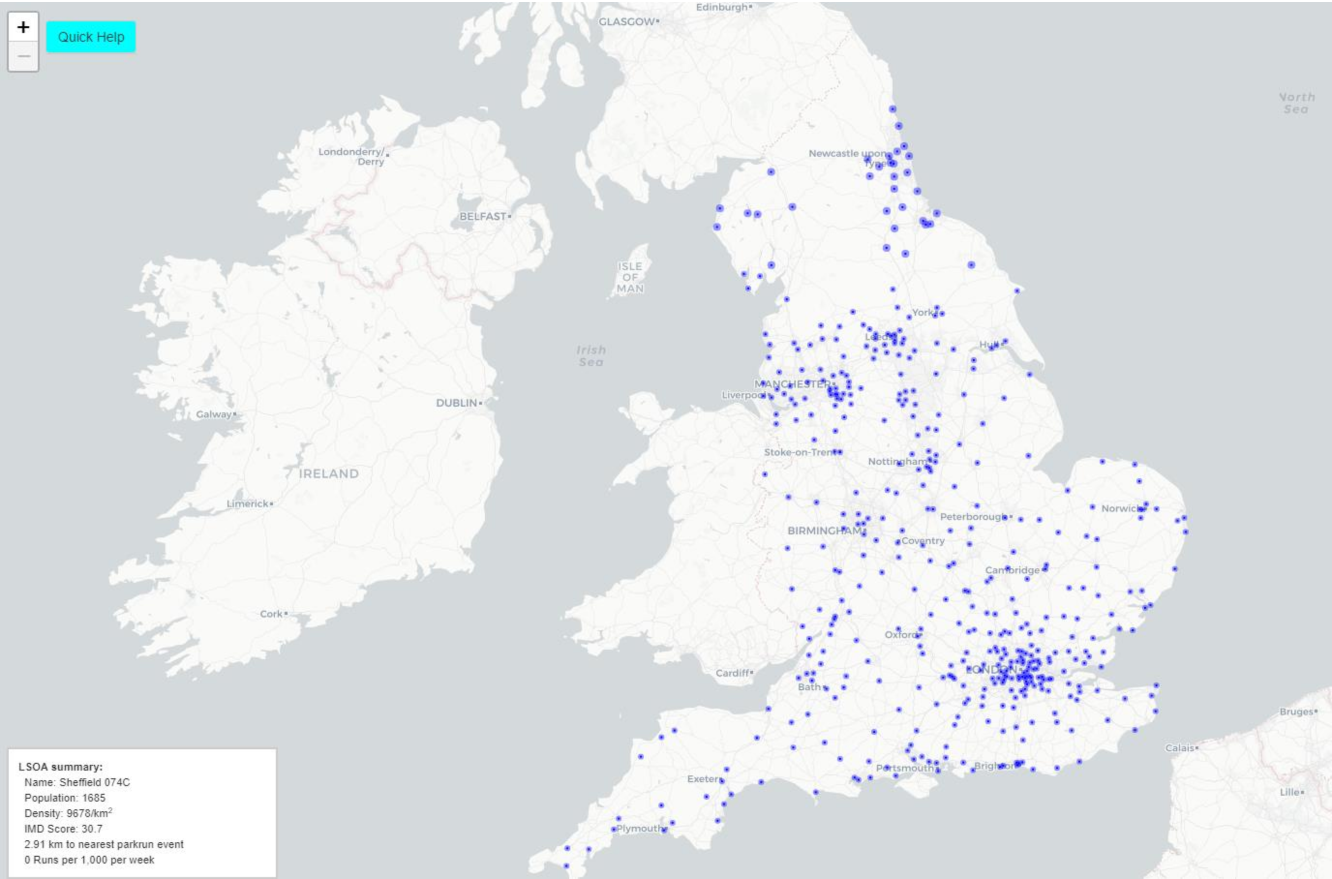
Despite the many strengths of the script based approach (e.g R) to decision modelling, an important limitation has been the lack of an easy-to-understand user-interface. While it is common practice for 'spreadsheet models' to have a structured front tab, which allows decision makers to manipulate model assumptions and

Who is teaching what?


Studio
DARTH
Input
Model
devtools
DARTH
ggplot2
ScHARR
School of Health and Related Research
Plots
ScHARR
School of Health and Related Research
Output
markdown
ScHARR
School of Health and Related Research
Shiny
Present

Examples

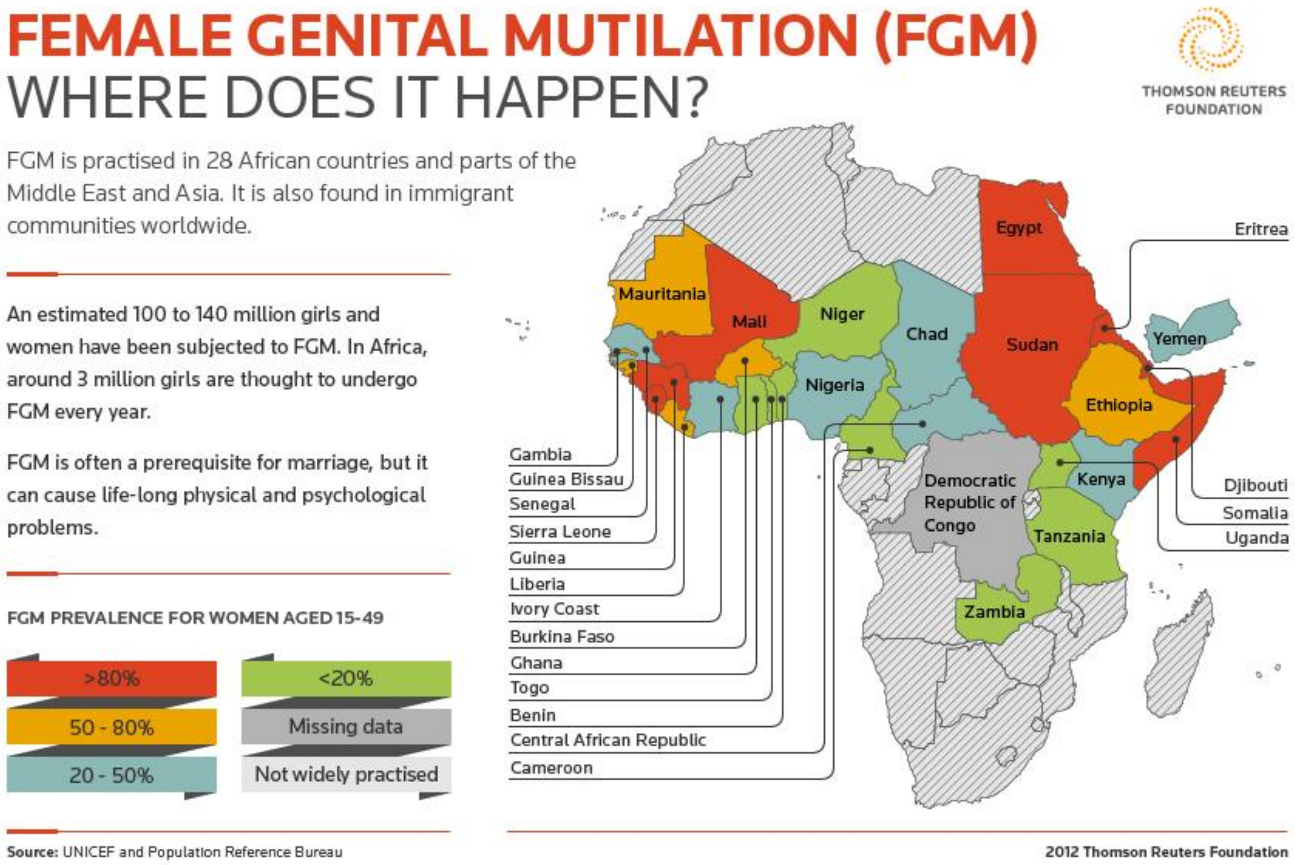
Where should parkrun locate new events?



LSOA summary:
Name: Sheffield 074C
Population: 1685
Density: 9678/km^2
IMD Score: 30.7
2.91 km to nearest parkrun event
0 Runs per 1,000 per week

http://iol-map.shef.ac.uk/

World Health Organization - FGM Model



FEMALE GENITAL MUTILATION (FGM)
WHERE DOES IT HAPPEN?

THOMSON REUTERS FOUNDATION

FGM is practised in 28 African countries and parts of the Middle East and Asia. It is also found in immigrant communities worldwide.

An estimated 100 to 140 million girls and women have been subjected to FGM. In Africa, around 3 million girls are thought to undergo FGM every year.

FGM is often a prerequisite for marriage, but it can cause life-long physical and psychological problems.

FGM PREVALENCE FOR WOMEN AGED 15-49

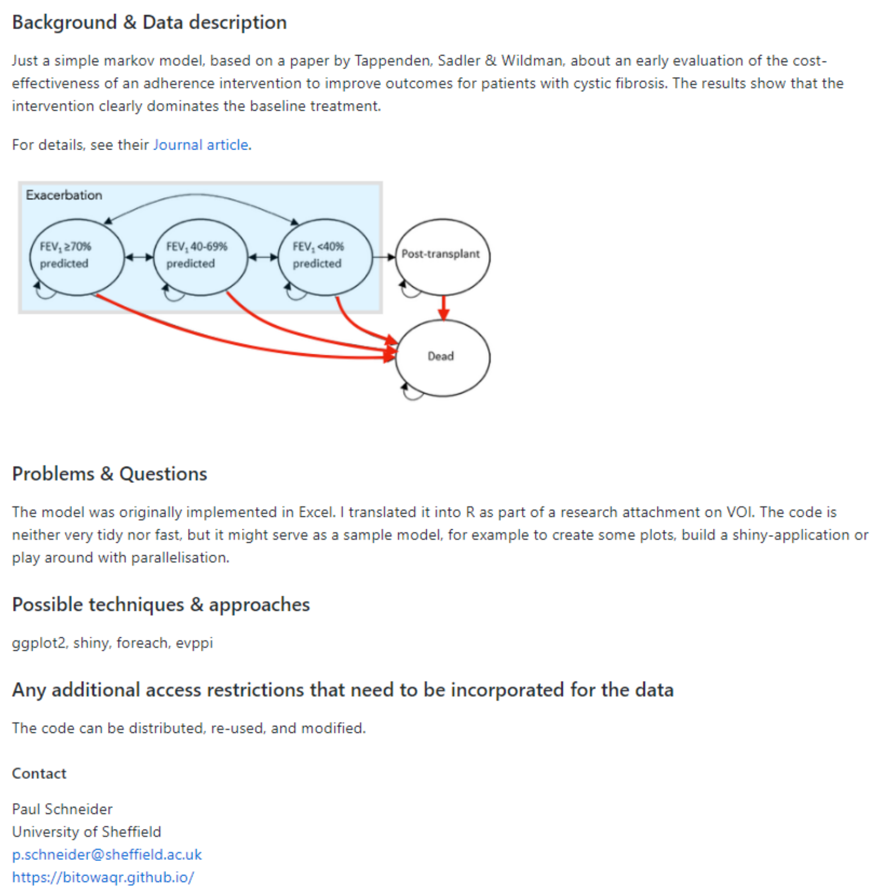
🟥 >80% 🟩 <20%
🟧 50 - 80% ⬜ Missing data
🟦 20 - 50% ▨ Not widely practised

Source: UNICEF and Population Reference Bureau

2012 Thomson Reuters Foundation

https://srhr.org/fgmcost/

Cystic Fibrosis Adherence



Background & Data description

Just a simple markov model, based on a paper by Tappenden, Sadler & Wildman, about an early evaluation of the cost-effectiveness of an adherence intervention to improve outcomes for patients with cystic fibrosis. The results show that the intervention clearly dominates the baseline treatment.

For details, see their Journal article.

Problems & Questions

The model was originally implemented in Excel. I translated it into R as part of a research attachment on VOI. The code is neither very tidy nor fast, but it might serve as a sample model, for example to create some plots, build a shiny-application or play around with parallelisation.

Possible techniques & approaches

ggplot2, shiny, foreach, evppi

Any additional access restrictions that need to be incorporated for the data

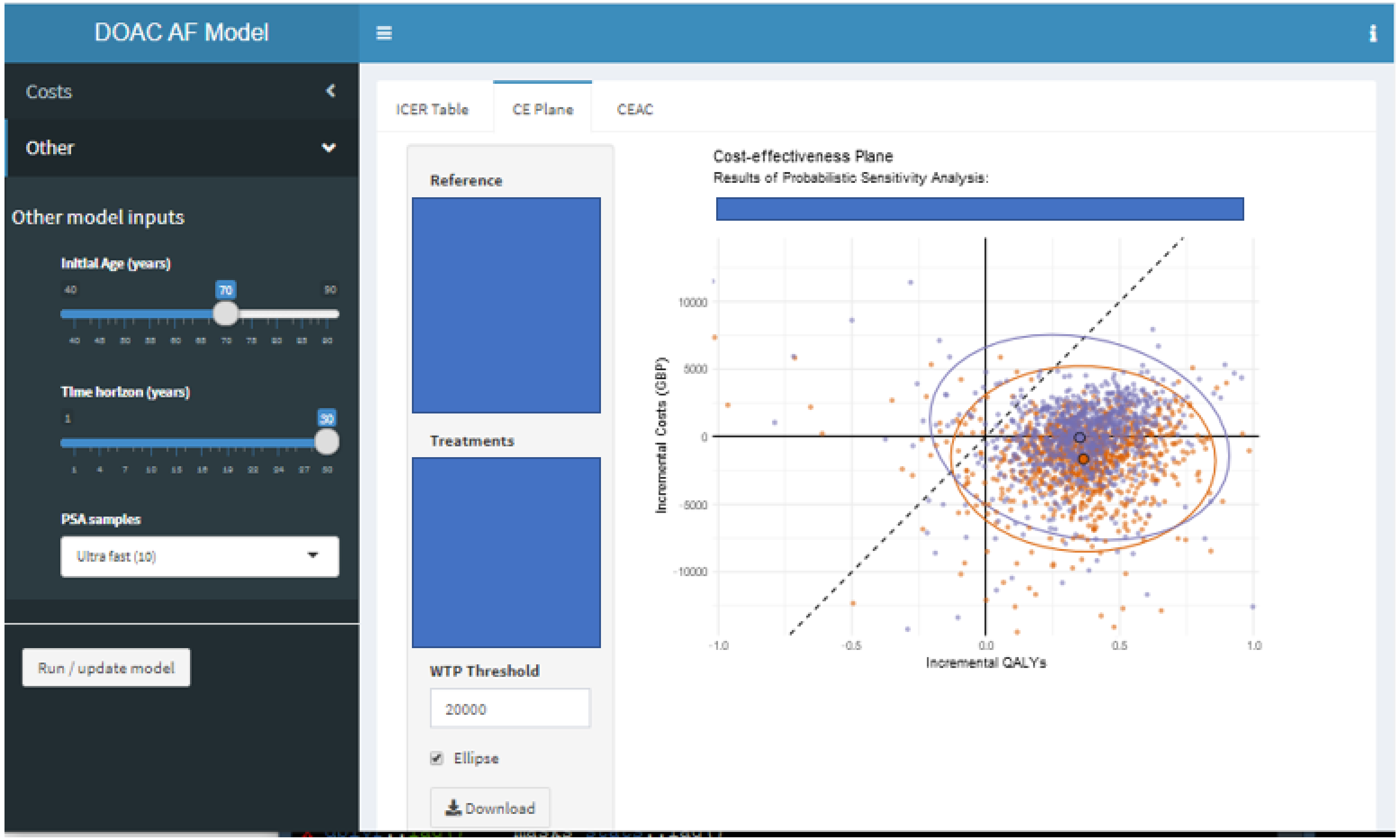
The code can be distributed, re-used, and modified.

Contact

Paul Schneider
University of Sheffield
p.schneider@sheffield.ac.uk
https://bitowaqr.github.io/

https://github.com/HealthEconomicsDataDive/shiny_s5

Health Technology Assessment



Initial Age (years): 70 (40–90)

Time horizon (years): 50 (1–50)

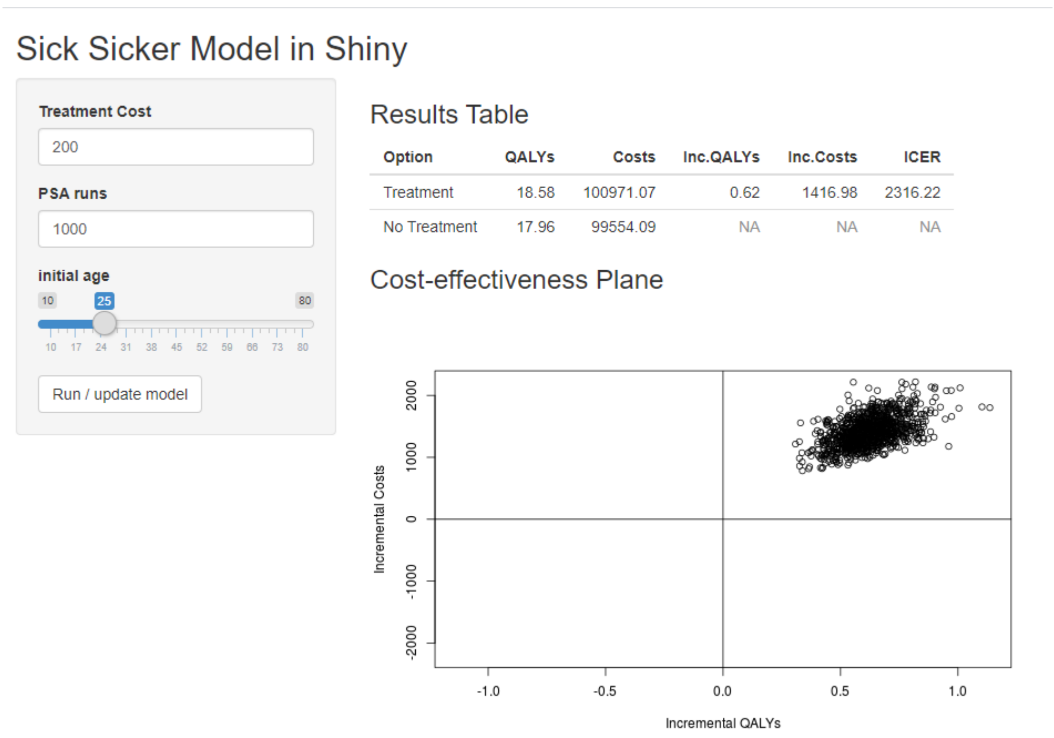
PSA samples: Ultra fast (10)

WTP Threshold: 20000

☑ Ellipse

Tutorial

Simple App



Sick Sicker Model in Shiny

Treatment Cost: 200

PSA runs: 1000

initial age: 25

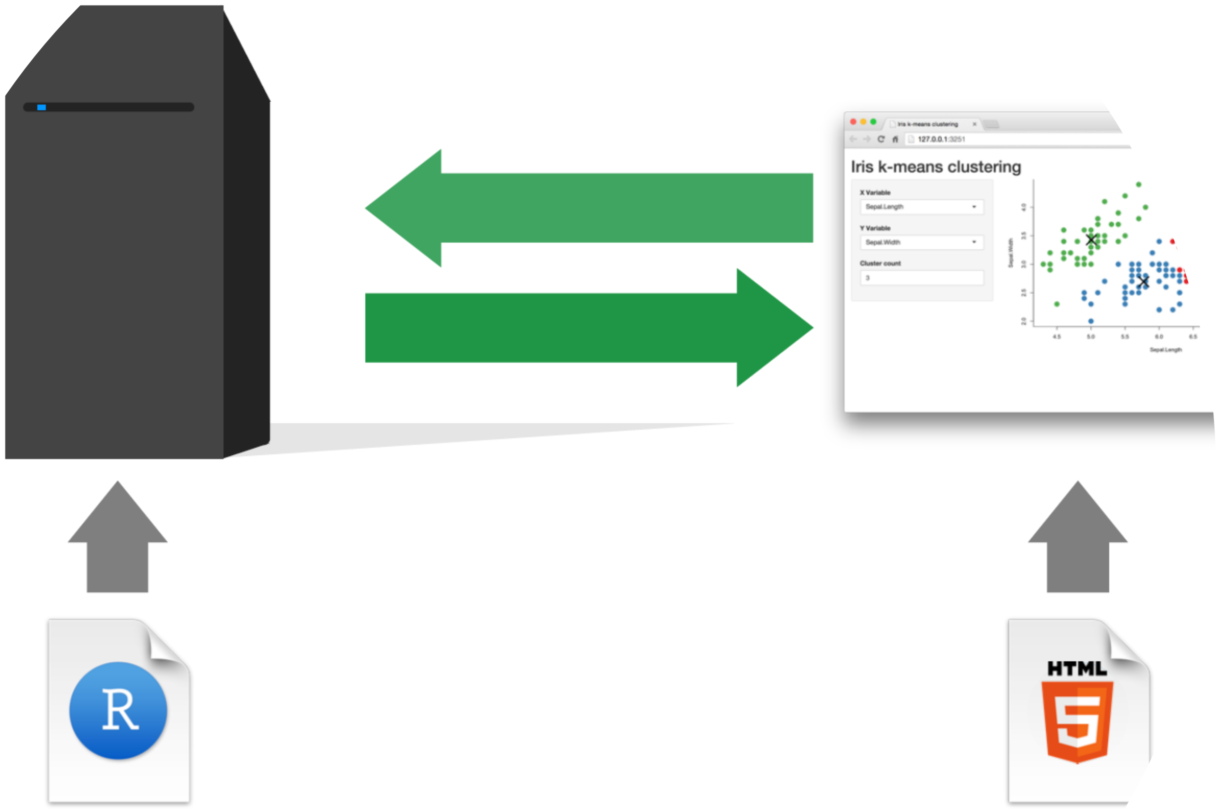
Results Table

Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.58	100971.07	0.62	1416.98	2316.22
No Treatment	17.96	99554.09	NA	NA	NA

Cost-effectiveness Plane

https://robertasmith.shinyapps.io/sick_sicker/

How does Shiny work?



Iris k-means clustering

X Variable

Sepal.Length

Y Variable

Sepal.Width

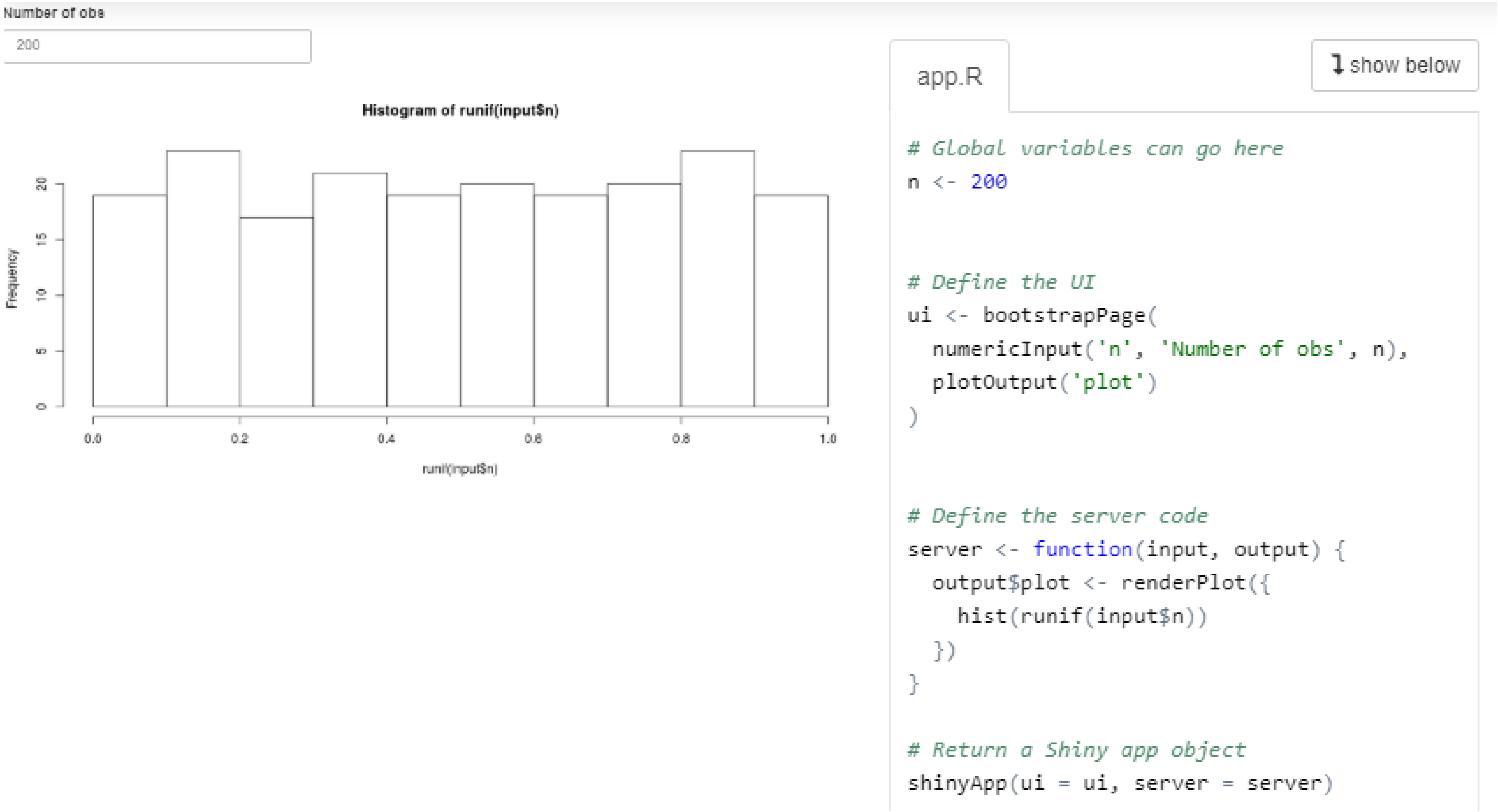
Cluster count

3

'er Instructions

User Inter'

Simple shiny application



```
# Global variables can go here
n <- 200


# Define the UI
ui <- bootstrapPage(
  numericInput('n', 'Number of obs', n),
  plotOutput('plot')
)


# Define the server code
server <- function(input, output) {
  output$plot <- renderPlot({
    hist(runif(input$n))
  })
}

# Return a Shiny app object
shinyApp(ui = ui, server = server)
```

https://shiny.rstudio.com/gallery/single-file-shiny-app.html

Cost-effectiveness model in R

Inputs

Parameters		
c_s1	cost1	3
c_s2	cost2	5
c_H	cost3	6
dr	Dis_rate	0.035
n_sim	No. psa	1000

Function

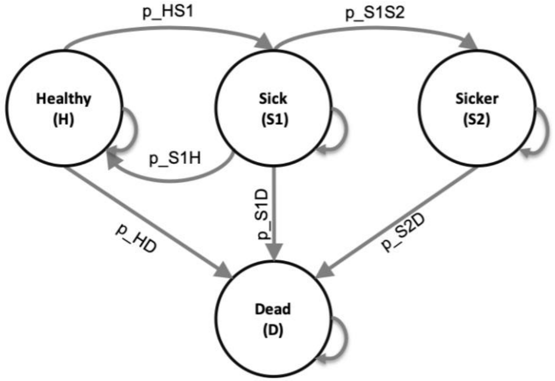
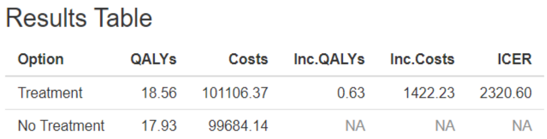


Figure 1: State-transition diagram of the time-independent Sick-Sicker cohort state-transition model with the name of the health states and possible transitions with their corresponding transition probabilities.

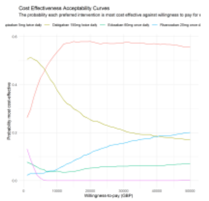
Outputs



Results Table

Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.56	101106.37	0.63	1422.23	2320.60
No Treatment	17.93	99684.14	NA	NA	NA

Cost-effectiveness Plane
Results of Probabilistic Sensitivity Analysis: Dabigatran (150mg bd) vs Coumarin (INR 2-3).
Incremental Costs (GBP)
Incremental QALYs



Steps

1. Understanding the model inputs and outputs.
2. Wrapping the model into a function.
3. Creating the app (iteratively adjusting user interface & server)
4. Deploying the app

Understanding the model inputs and outputs

DARTH Sick Sicker Model

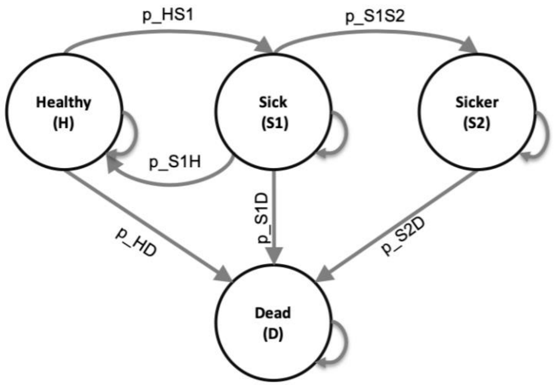
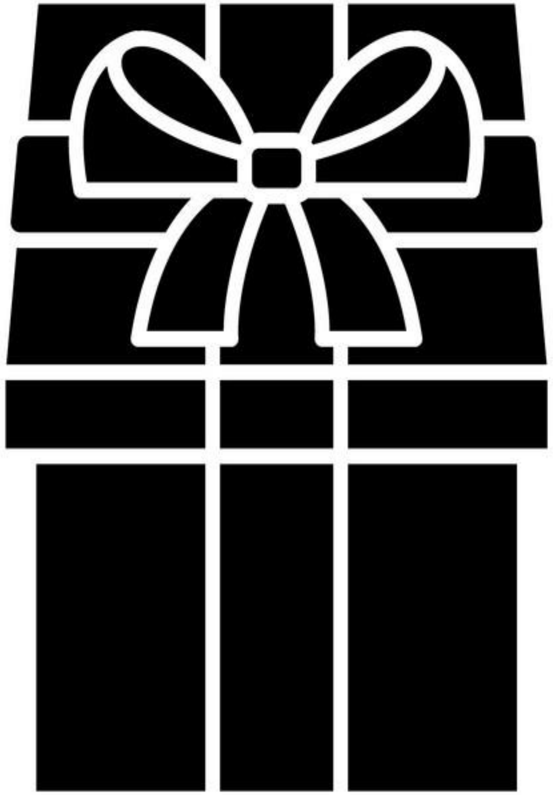


Figure 1: State-transition diagram of the time-independent Sick-Sicker cohort state-transition model with the name of the health states and possible transitions with their corresponding transition probabilities.

https://github.com/RobertASmith/healthecon_shiny/tree/master/Tutorial

Wrapping the model into a function



Wrapping the model into a function

Inputs

Function

Outputs

Parameters		
c_s1	cost1	3
c_s2	cost2	5
c_H	cost3	6
dr	Dis_rate	0.035
n_sim	No. psa	1000

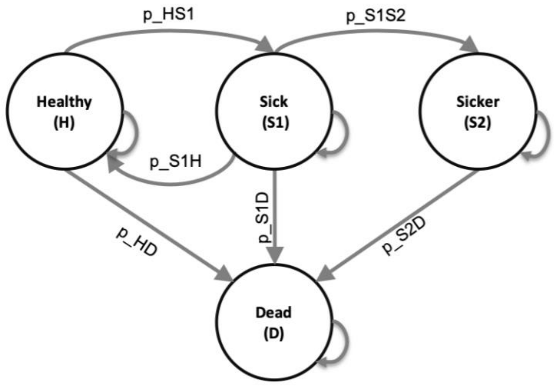
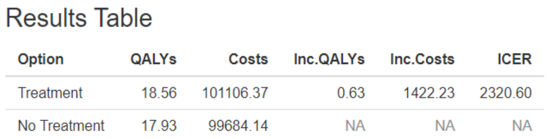
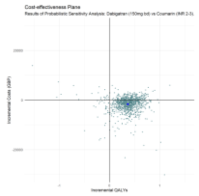


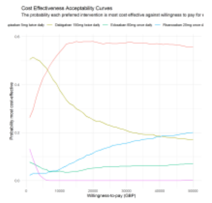
Figure 1: State-transition diagram of the time-independent Sick-Sicker cohort state-transition model with the name of the health states and possible transitions with their corresponding transition probabilities.



Results Table

Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.56	101106.37	0.63	1422.23	2320.60
No Treatment	17.93	99684.14	NA	NA	NA

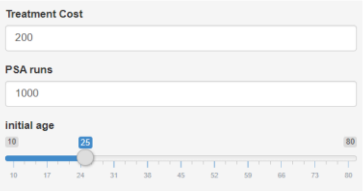




Cost-effectiveness model in R-shiny

Inputs → Function → Outputs

Parameters		
c_s1	cost1	3
c_s2	cost2	5
c_H	cost3	6
dr	Dis_rate	0.035
n_sim	No. psa	1000



Treatment Cost

200

PSA runs

1000

initial age

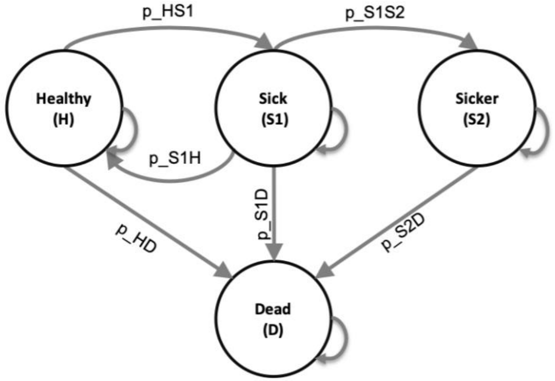
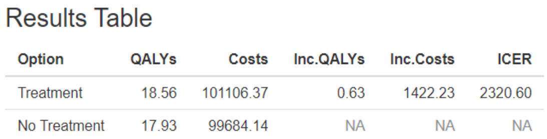
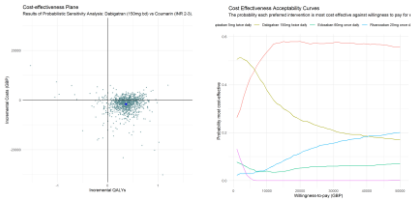


Figure 1: State-transition diagram of the time-independent Sick-Sicker cohort state-transition model with the name of the health states and possible transitions with their corresponding transition probabilities.



Results Table

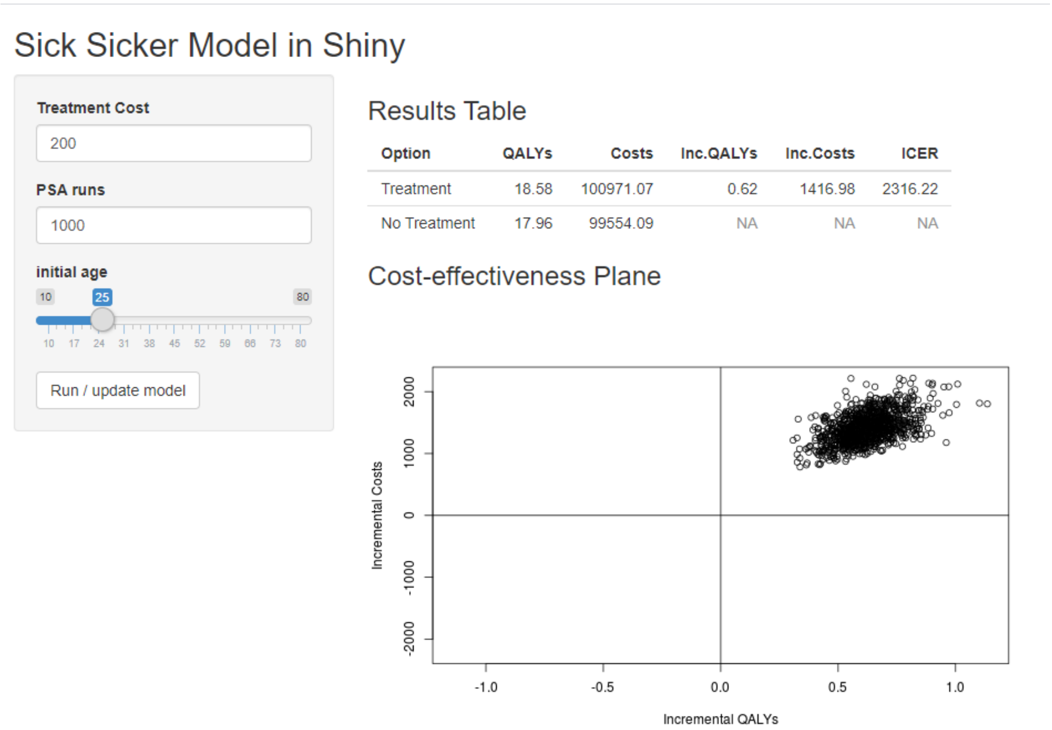
Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.56	101106.37	0.63	1422.23	2320.60
No Treatment	17.93	99684.14	NA	NA	NA



Creating the app

User Interface

Server



Sick Sicker Model in Shiny

Treatment Cost

200

PSA runs

1000

initial age

Results Table

Option	QALYs	Costs	Inc.QALYs	Inc.Costs	ICER
Treatment	18.58	100971.07	0.62	1416.98	2316.22
No Treatment	17.96	99554.09	NA	NA	NA

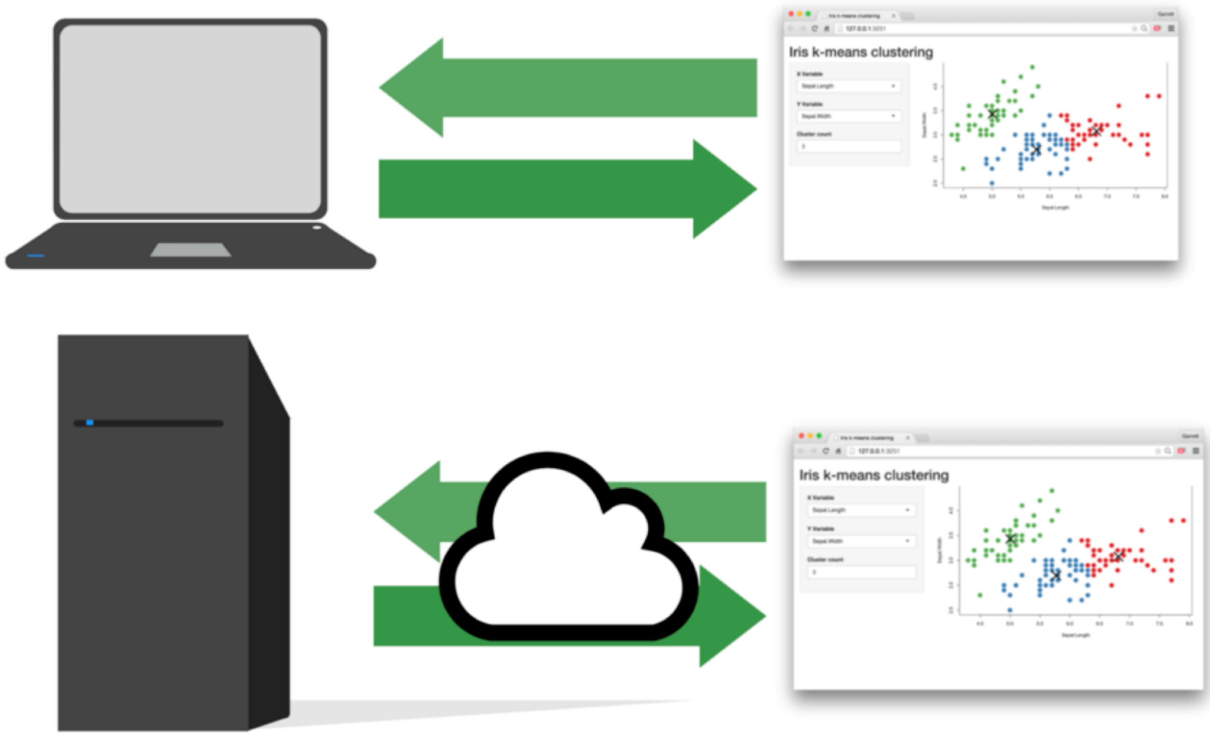
Cost-effectiveness Plane

https://github.com/RobertASmith/healthecon_shiny/blob/master/App/app.R

Deploying the app

Whether to upload to the cloud?

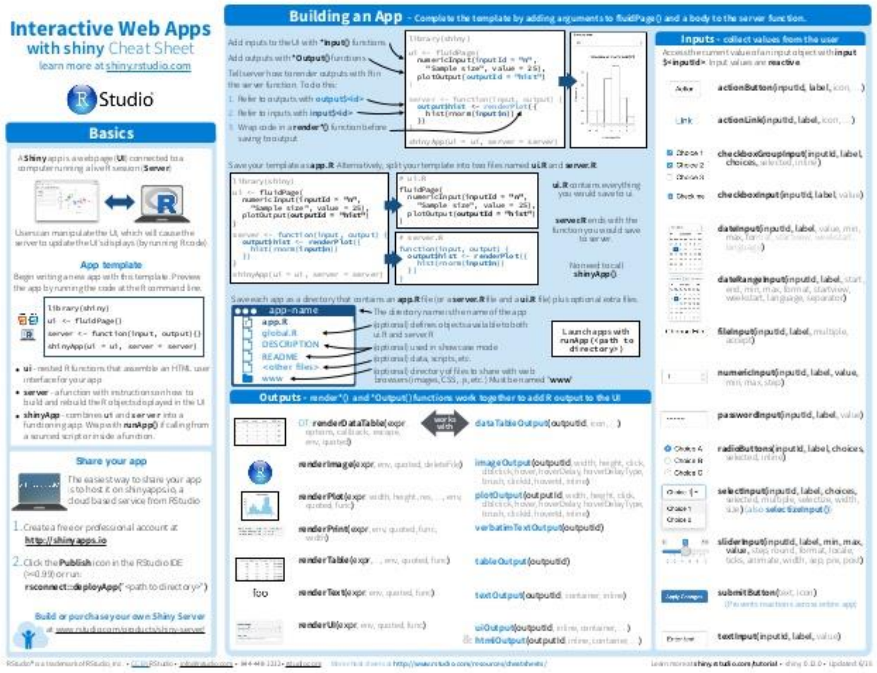
Server (running R) *Client* (draws user interface)



Local… both are the same machine, probably your laptop

Cloud… server is somewhere else on the web and the client is the user’s machine

Further help …



Interactive Web Apps with shiny Cheat Sheet

learn more at shiny.rstudio.com

RStudio

Basics

A **Shiny** app is a web page (**UI**) connected to a computer running a live R session (**Server**)

Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

App template

Begin writing a new app with this template. Preview the app by running the code at the R command line.

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

- **ui** - nested R functions that assemble an HTML user interface for your app
- **server** - a function with instructions on how to build and rebuild the R objects displayed in the UI
- **shinyApp** - combines **ui** and **server** into a functioning app. Wrap with **runApp()** if calling from a sourced script or inside a function.

Share your app

The easiest way to share your app is to host it on shinyapps.io, a cloud based service from RStudio

1. Create a free or professional account at http://shinyapps.io
2. Click the **Publish** icon in the RStudio IDE (>=0.99) or run: **rsconnect::deployApp**("<path to directory>")

Build or purchase your own Shiny Server at www.rstudio.com/products/shiny-server/

Building an App - Complete the template by adding arguments to fluidPage() and a body to the server function.

Add inputs to the UI with ***Input()** functions

Add outputs with ***Output()** functions

Tell server how to render outputs with R in the server function. To do this:

1. Refer to outputs with **output$<id>**
2. Refer to inputs with **input$<id>**
3. Wrap code in a **render*()** function before saving to output

```
library(shiny)
ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
shinyApp(ui = ui, server = server)
```

Save your template as **app.R**. Alternatively, split your template into two files named **ui.R** and **server.R**.

```
# ui.R
fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
```

```
# server.R
function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
```

ui.R contains everything you would save to ui.

server.R ends with the function you would save to server.

No need to call **shinyApp()**.

Save each app as a directory that contains an **app.R** file (or a **server.R** file and a **ui.R** file) plus optional extra files.

app-name ← The directory name is the name of the app
- app.R
- global.R ← (optional) defines objects available to both ui.R and server.R
- DESCRIPTION ← (optional) used in showcase mode
- README ← (optional) data, scripts, etc.
- <other files> ← (optional) directory of files to share with web browsers (images, CSS, .js, etc.) Must be named "www"
- www

Launch apps with runApp(<path to directory>)

Outputs - render*() and *Output() functions work together to add R output to the UI

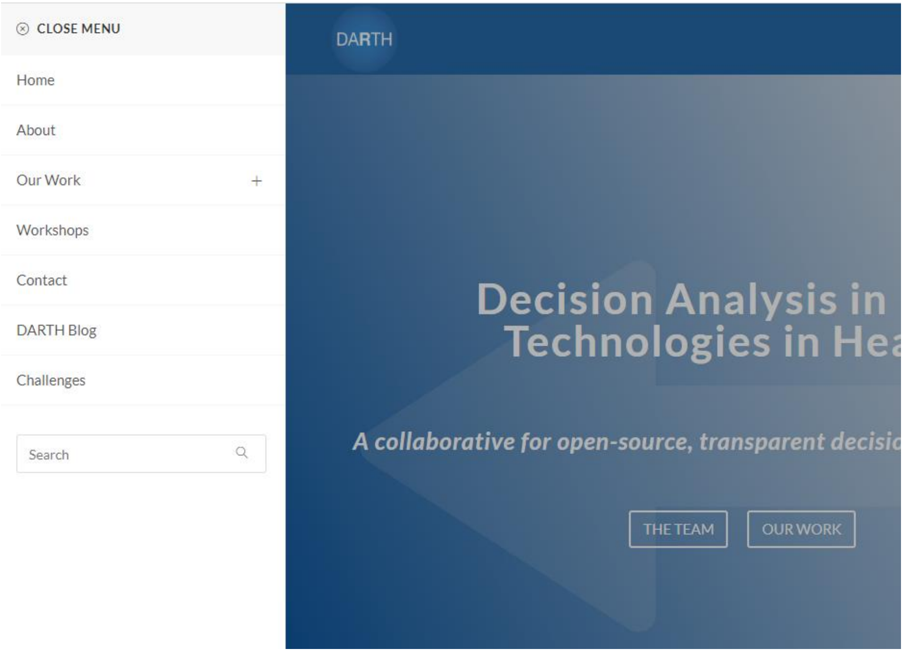
	works with	
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)		**dataTableOutput**(outputId, icon, ...)
renderImage(expr, env, quoted, deleteFile)		**imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)
renderPlot(expr, width, height, res, ..., env, quoted, func)		**plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)
renderPrint(expr, env, quoted, func, width)		**verbatimTextOutput**(outputId)
renderTable(expr, ..., env, quoted, func)		**tableOutput**(outputId)
renderText(expr, env, quoted, func)		**textOutput**(outputId, container, inline)
renderUI(expr, env, quoted, func)		**uiOutput**(outputId, inline, container, ...) & **htmlOutput**(outputId, inline, container, ...)

Inputs - collect values from the user

Access the current value of an input object with **input$<inputId>**. Input values are **reactive**.

- **actionButton**(inputId, label, icon, ...)
- **actionLink**(inputId, label, icon, ...)
- **checkboxGroupInput**(inputId, label, choices, selected, inline)
- **checkboxInput**(inputId, label, value)
- **dateInput**(inputId, label, value, min, max, format, startview, weekstart, language)
- **dateRangeInput**(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)
- **fileInput**(inputId, label, multiple, accept)
- **numericInput**(inputId, label, value, min, max, step)
- **passwordInput**(inputId, label, value)
- **radioButtons**(inputId, label, choices, selected, inline)
- **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)
- **sliderInput**(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)
- **submitButton**(text, icon) (Prevents reactions across entire app)
- **textInput**(inputId, label, value)

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com More cheat sheets at http://www.rstudio.com/resources/cheatsheets/ Learn more at shiny.rstudio.com/tutorial • shiny 0.12.0 • Updated 6/15



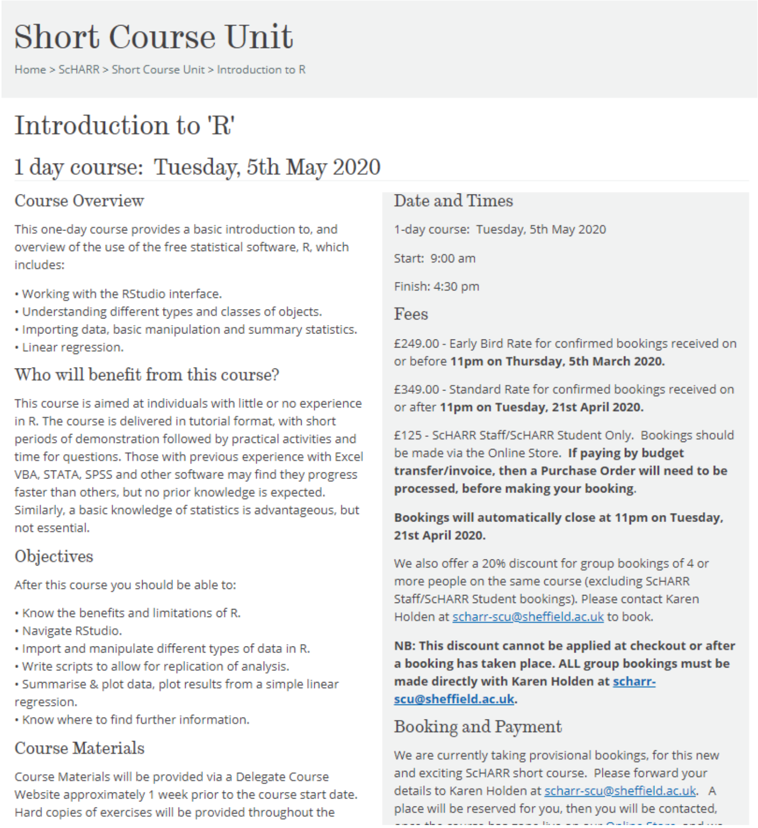
Decision Analysis in Technologies in Hea

A collaborative for open-source, transparent decisio

https://blog.rstudio.com/2014/06/30/shiny-cheat-sheet/

http://darthworkgroup.com/

Shameless self promotion



Short Course Unit

Home > ScHARR > Short Course Unit > Introduction to R

Introduction to 'R'

1 day course: Tuesday, 5th May 2020

Course Overview

This one-day course provides a basic introduction to, and overview of the use of the free statistical software, R, which includes:

- Working with the RStudio interface.
- Understanding different types and classes of objects.
- Importing data, basic manipulation and summary statistics.
- Linear regression.

Who will benefit from this course?

This course is aimed at individuals with little or no experience in R. The course is delivered in tutorial format, with short periods of demonstration followed by practical activities and time for questions. Those with previous experience with Excel VBA, STATA, SPSS and other software may find they progress faster than others, but no prior knowledge is expected. Similarly, a basic knowledge of statistics is advantageous, but not essential.

Objectives

After this course you should be able to:

- Know the benefits and limitations of R.
- Navigate RStudio.
- Import and manipulate different types of data in R.
- Write scripts to allow for replication of analysis.
- Summarise & plot data, plot results from a simple linear regression.
- Know where to find further information.

Course Materials

Course Materials will be provided via a Delegate Course Website approximately 1 week prior to the course start date. Hard copies of exercises will be provided throughout the

Date and Times

1-day course: Tuesday, 5th May 2020

Start: 9:00 am

Finish: 4:30 pm

Fees

£249.00 - Early Bird Rate for confirmed bookings received on or before **11pm on Thursday, 5th March 2020.**

£349.00 - Standard Rate for confirmed bookings received on or after **11pm on Tuesday, 21st April 2020.**

£125 - ScHARR Staff/ScHARR Student Only. Bookings should be made via the Online Store. **If paying by budget transfer/invoice, then a Purchase Order will need to be processed, before making your booking.**

Bookings will automatically close at 11pm on Tuesday, 21st April 2020.

We also offer a 20% discount for group bookings of 4 or more people on the same course (excluding ScHARR Staff/ScHARR Student bookings). Please contact Karen Holden at scharr-scu@sheffield.ac.uk to book.

NB: This discount cannot be applied at checkout or after a booking has taken place. ALL group bookings must be made directly with Karen Holden at scharr-scu@sheffield.ac.uk.

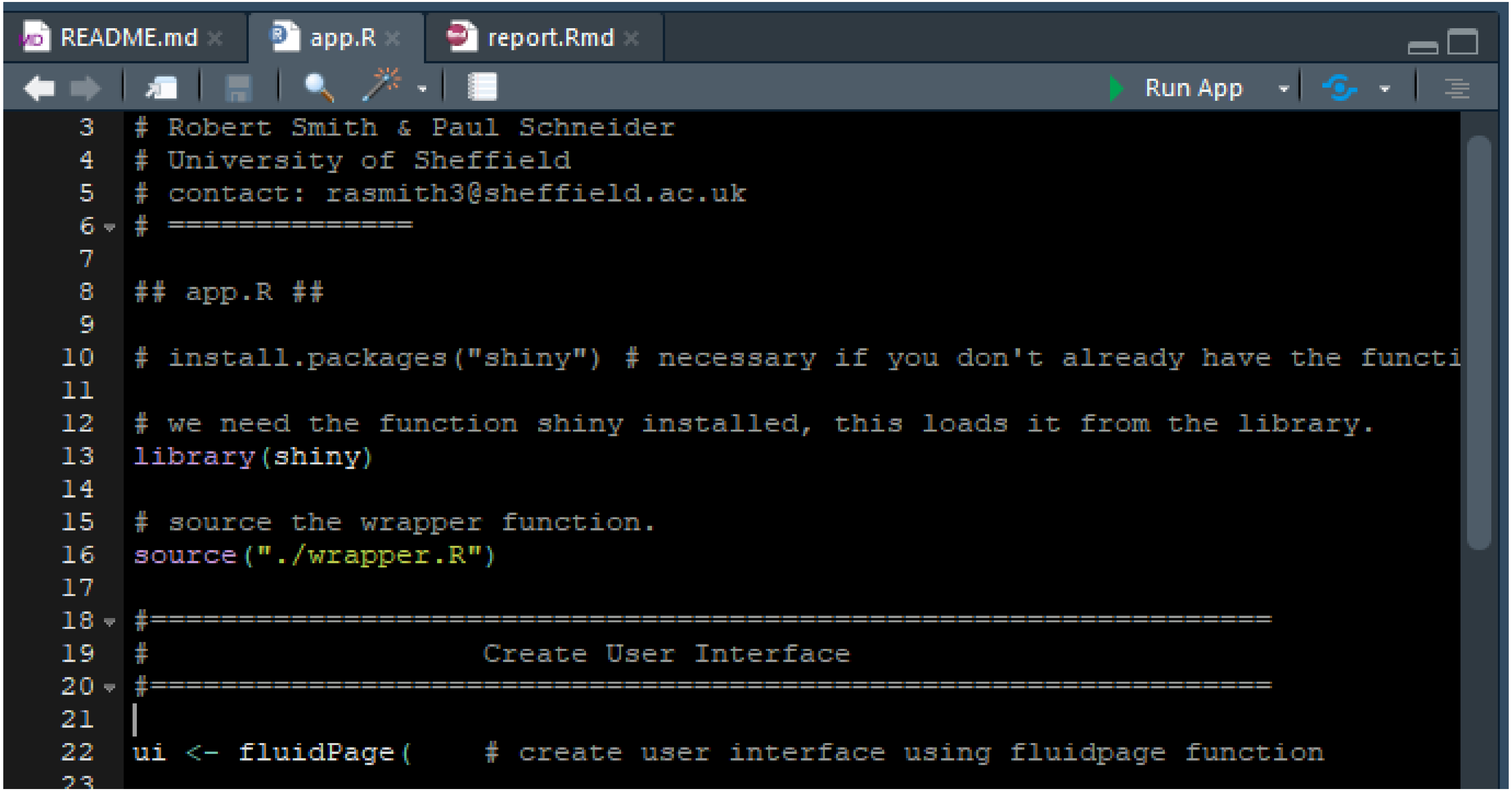
Booking and Payment

We are currently taking provisional bookings, for this new and exciting ScHARR short course. Please forward your details to Karen Holden at scharr-scu@sheffield.ac.uk.   A place will be reserved for you, then you will be contacted, once the course has gone live on our Online Store, and we

R and Shiny for Health Economics?

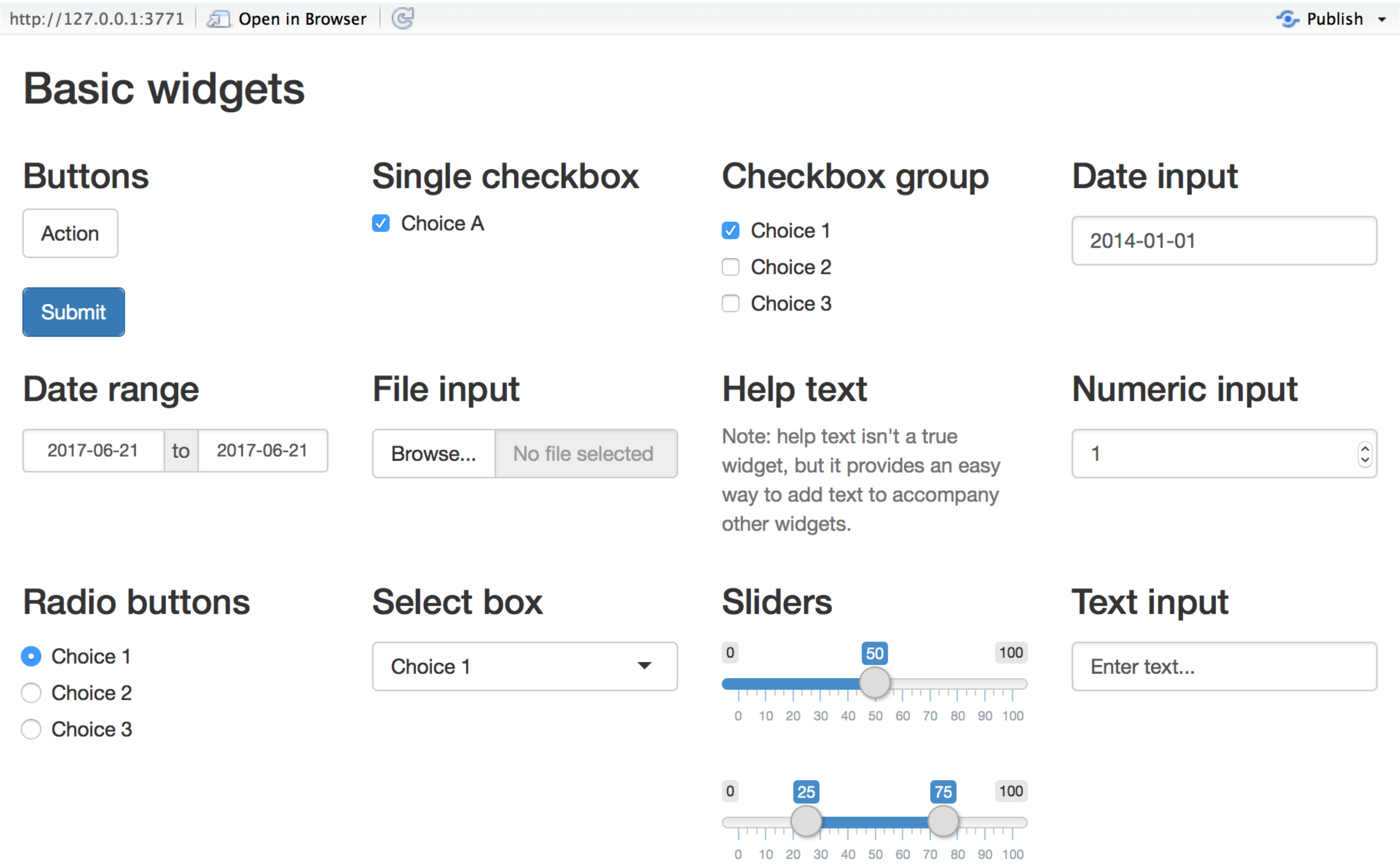
https://www.sheffield.ac.uk/scharr/shortcourseunit/intror

Extra Slides



```
# Robert Smith & Paul Schneider
# University of Sheffield
# contact: rasmith3@sheffield.ac.uk
# ==============

## app.R ##

# install.packages("shiny") # necessary if you don't already have the functi

# we need the function shiny installed, this loads it from the library.
library(shiny)

# source the wrapper function.
source("./wrapper.R")

#================================================================
#                    Create User Interface
#================================================================

ui <- fluidPage(    # create user interface using fluidpage function
```

Inputs



Basic widgets

Buttons

Single checkbox

☑ Choice A

Checkbox group

☑ Choice 1
□ Choice 2
□ Choice 3

Date input

2014-01-01

Date range

2017-06-21 to 2017-06-21

File input

No file selected

Help text

Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Numeric input

1

Radio buttons

● Choice 1
○ Choice 2
○ Choice 3

Select box

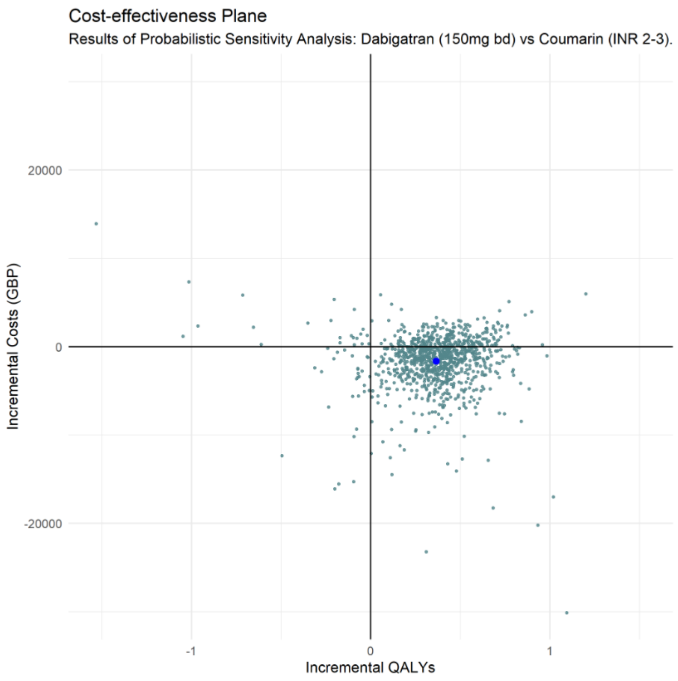
Choice 1

Sliders

Text input

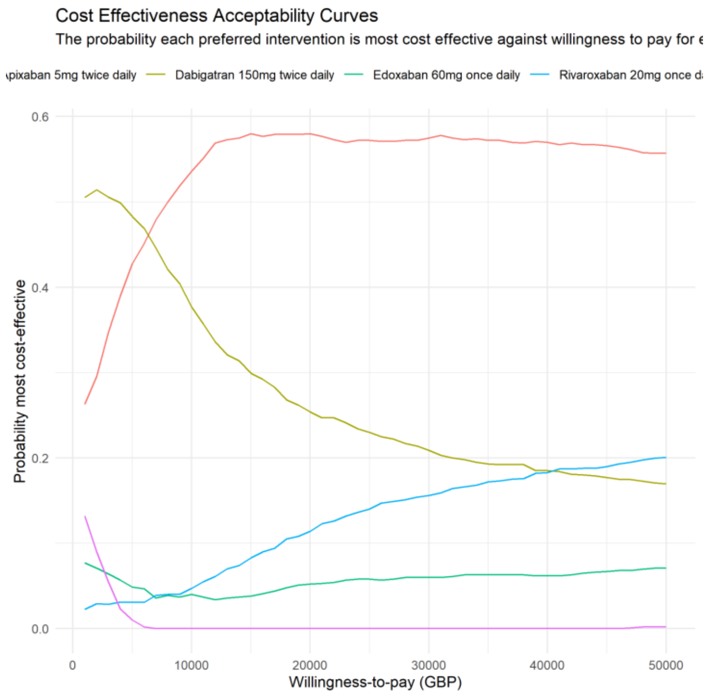
Enter text...

Outputs


Cost-effectiveness Plane

Results of Probabilistic Sensitivity Analysis: Dabigatran (150mg bd) vs Coumarin (INR 2-3).

ggplot2


Cost Effectiveness Acceptability Curves

The probability each preferred intervention is most cost effective against willingness to pay for e

Coding framework

Decision Analysis in R for Technologies in Health

A collaborative for open-source, transparent decision analytic modelling

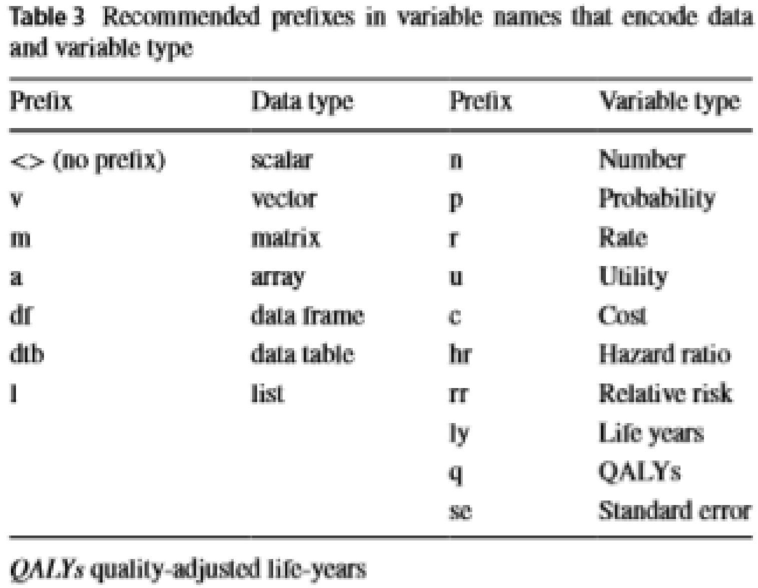


Table 3 Recommended prefixes in variable names that encode data and variable type

Prefix	Data type	Prefix	Variable type
<> (no prefix)	scalar	n	Number
v	vector	p	Probability
m	matrix	r	Rate
a	array	u	Utility
df	data frame	c	Cost
dtb	data table	hr	Hazard ratio
l	list	rr	Relative risk
		ly	Life years
		q	QALYs
		se	Standard error

QALYs quality-adjusted life-years

Alarid-Escudero, F., Krijkamp, E.M., Pechlivanoglou, P., Jalal, H., Kao, S.Y.Z., Yang, A. and Enns, E.A., 2019. A need for change! A coding framework for improving transparency in decision modeling. *PharmacoEconomics*, *37*(11), pp.1329-1339.

Sourcing functions outside of wrapper

- Inner function can call from global function and itself, not functions above.

- environment(noac.net.benefit) <- environment() ~ specifies that the environment for this function is the same as the function that this line is within!!!

Run button

SERVER

```
server <- function(input, output,) {

observeEvent(input$run_model, ignoreNULL = F, {

model_results =  f_model_wrapper(input1 = input$slider1)

              } # close observe event
} # close server function
```

UI

```
   ui <-     dashboardPage(

                        …

  actionButton("run_model","Run / update model")

                        …

)
```

Visualization functions

Plotting the CE-Plane & CEAC

Plot visualizations

SERVER

```
output$CE_plane <- renderPlot({


plot_cep( model_output = model_res()$model.outputs)


  })
```

UI

```
tabPanel("CE Plane",

plotOutput("CE_plane",height = 450),

fluidPage(downloadButton('cep')))
```

plot_cep function

```
plot_cep <- function(model_output) ){

<insert code to get results into right format, incremental costs in long format vs comparitor>

ce_plane_plot = ggplot()  +
geom_point(data= incr_long, aes(x=incr_Q, y=incr_C, col=Treatment)) +
geom_vline(xintercept = 0) +
   geom_hline(yintercept = 0) +
   theme_minimal()  +
   labs(title = "Cost-effectiveness Plane",
       subtitle = paste0("Results of Probabilistic Sensitivity Analysis:")) +
   # labels
   xlab("Incremental QALYs") +
   ylab("Incremental Costs (GBP)")



return(ce_plane_plot)
}
```

Download plot functionality

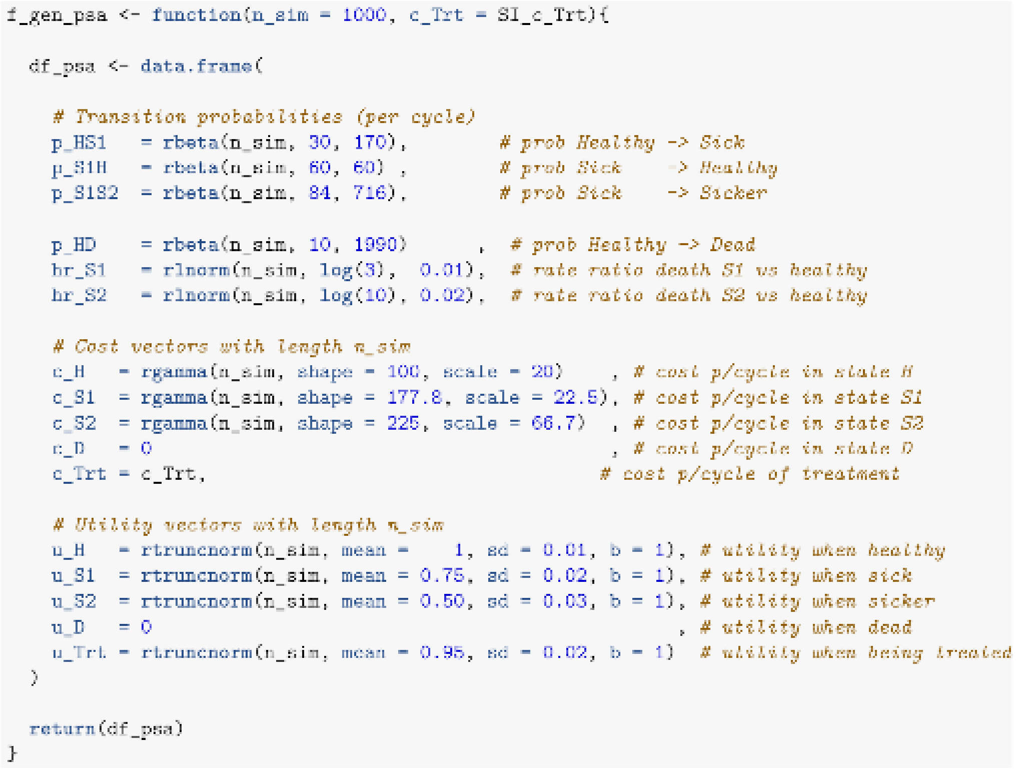
SERVER

```
ceP_download <- plot_ce.plane.plot(model_output = model_res()$model.outputs)


  output$cep = downloadHandler(
    filename = 'ce_plane.png',
    content = function(file) {
      device <- function(..., width, height) {
        grDevices::png(..., width = width, height = height,
                      res = 300, units = "in")
      }
      ggsave(file,
             plot = ceP_download,
            device = device)
    })
```

UI

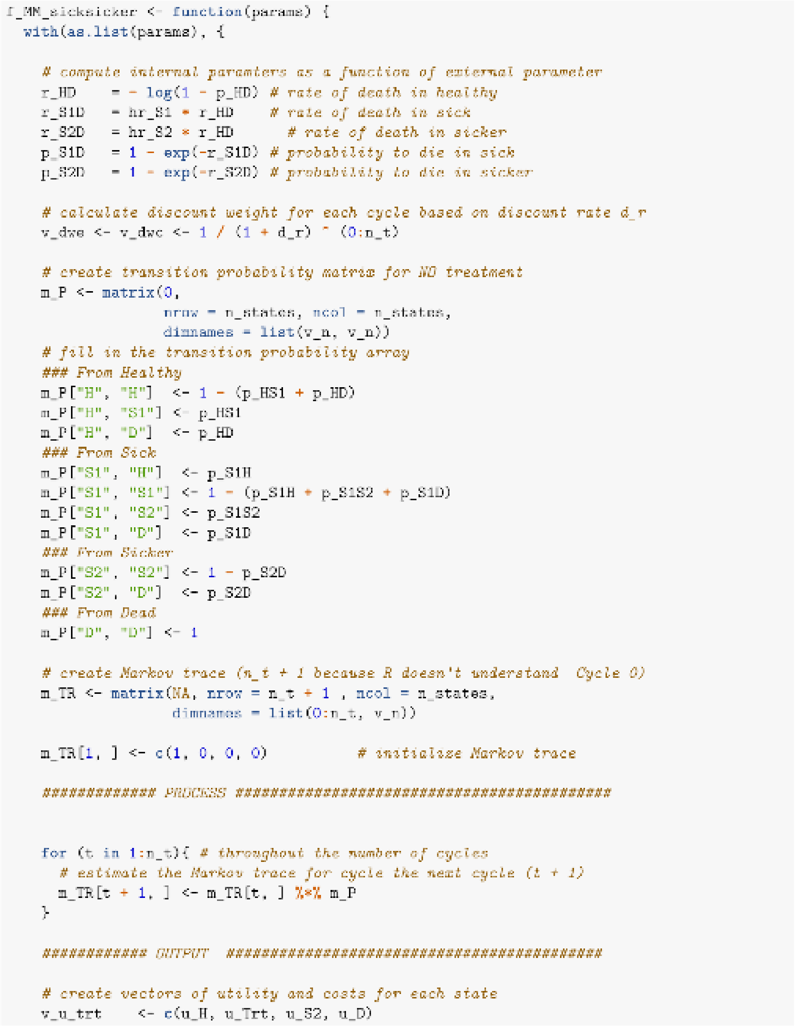
```
tabPanel("CE Plane",

plotOutput("CE_plane",height = 450),

fluidPage(downloadButton('cep')))
```

DARTH Sick-Sicker Model



```r
f_gen_psa <- function(n_sim = 1000, c_Trt = SI_c_Trt){

  df_psa <- data.frame(

    # Transition probabilities (per cycle)
    p_HS1   = rbeta(n_sim, 30, 170),        # prob Healthy -> Sick
    p_S1H   = rbeta(n_sim, 60, 60) ,        # prob Sick    -> Healthy
    p_S1S2  = rbeta(n_sim, 84, 716),        # prob Sick    -> Sicker

    p_HD    = rbeta(n_sim, 10, 1990)      ,  # prob Healthy -> Dead
    hr_S1   = rlnorm(n_sim, log(3),  0.01),  # rate ratio death S1 vs healthy
    hr_S2   = rlnorm(n_sim, log(10), 0.02),  # rate ratio death S2 vs healthy

    # Cost vectors with length n_sim
    c_H   = rgamma(n_sim, shape = 100, scale = 20)    , # cost p/cycle in state H
    c_S1  = rgamma(n_sim, shape = 177.8, scale = 22.5), # cost p/cycle in state S1
    c_S2  = rgamma(n_sim, shape = 225, scale = 66.7)  , # cost p/cycle in state S2
    c_D   = 0                                         , # cost p/cycle in state D
    c_Trt = c_Trt,                                     # cost p/cycle of treatment

    # Utility vectors with length n_sim
    u_H   = rtruncnorm(n_sim, mean =    1, sd = 0.01, b = 1), # utility when healthy
    u_S1  = rtruncnorm(n_sim, mean = 0.75, sd = 0.02, b = 1), # utility when sick
    u_S2  = rtruncnorm(n_sim, mean = 0.50, sd = 0.03, b = 1), # utility when sicker
    u_D   = 0                                               , # utility when dead
    u_Trt = rtruncnorm(n_sim, mean = 0.95, sd = 0.02, b = 1)  # utility when being treated
  )

  return(df_psa)
}
```

DARTH Sick-Sicker Model



```r
f_MM_sicksicker <- function(params) {
  with(as.list(params), {

    # compute internal paramters as a function of external parameter
    r_HD    = - log(1 - p_HD) # rate of death in healthy
    r_S1D   = hr_S1 * r_HD    # rate of death in sick
    r_S2D   = hr_S2 * r_HD      # rate of death in sicker
    p_S1D   = 1 - exp(-r_S1D) # probability to die in sick
    p_S2D   = 1 - exp(-r_S2D) # probability to die in sicker

    # calculate discount weight for each cycle based on discount rate d_r
    v_dwe <- v_dwc <- 1 / (1 + d_r) ^ (0:n_t)

    # create transition probability matrix for NO treatment
    m_P <- matrix(0,
                  nrow = n_states, ncol = n_states,
                  dimnames = list(v_n, v_n))
    # fill in the transition probability array
    ### From Healthy
    m_P["H", "H"]  <- 1 - (p_HS1 + p_HD)
    m_P["H", "S1"] <- p_HS1
    m_P["H", "D"]  <- p_HD
    ### From Sick
    m_P["S1", "H"]  <- p_S1H
    m_P["S1", "S1"] <- 1 - (p_S1H + p_S1S2 + p_S1D)
    m_P["S1", "S2"] <- p_S1S2
    m_P["S1", "D"]  <- p_S1D
    ### From Sicker
    m_P["S2", "S2"] <- 1 - p_S2D
    m_P["S2", "D"]  <- p_S2D
    ### From Dead
    m_P["D", "D"] <- 1

    # create Markov trace (n_t + 1 because R doesn't understand  Cycle 0)
    m_TR <- matrix(NA, nrow = n_t + 1 , ncol = n_states,
                   dimnames = list(0:n_t, v_n))

    m_TR[1, ] <- c(1, 0, 0, 0)          # initialize Markov trace

    ############# PROCESS ###########################################


    for (t in 1:n_t){ # throughout the number of cycles
      # estimate the Markov trace for cycle the next cycle (t + 1)
      m_TR[t + 1, ] <- m_TR[t, ] %*% m_P
    }

    ############ OUTPUT  ############################################

    # create vectors of utility and costs for each state
    v_u_trt    <- c(u_H, u_Trt, u_S2, u_D)
```

Basic code for model wrapper

```
f_wrapper =
        function(input1 = x,input2 = y, input3 = z){

                < insert model code reliant on input 1-3 >

                return(results)   # return model results

        } # close function bracket
```

Basic code for app

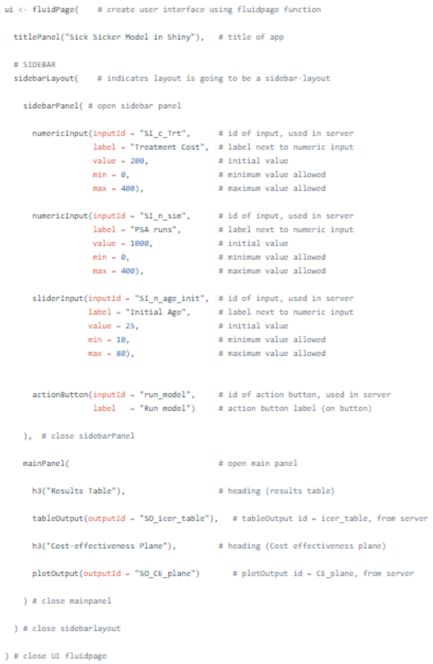
SERVER

```
server <- function(input, output,) {

model_results =  f_model_wrapper(input1 = input$slider1)

output$icer_table <- renderDataTable({
        < ggplot results table code or function>
    })

}
```

UI

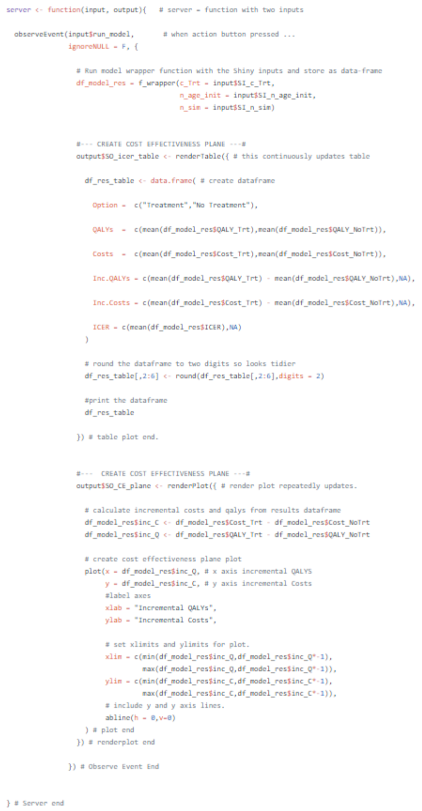
```
        ui <- dashboardPage(

sliderInput(inputId = “slider1",  label = “input1")

dataTableOutput("icer_table")

)
```

User Interface



```
ui <- fluidPage(    # create user interface using fluidpage function

  titlePanel("Sick Sicker Model in Shiny"),   # title of app

  # SIDEBAR
  sidebarLayout(    # indicates layout is going to be a sidebar-layout

    sidebarPanel( # open sidebar panel

      numericInput(inputId = "SI_c_Trt",      # id of input, used in server
                   label = "Treatment Cost",  # label next to numeric input
                   value = 200,               # initial value
                   min = 0,                   # minimum value allowed
                   max = 400),                # maximum value allowed

      numericInput(inputId = "SI_n_sim",      # id of input, used in server
                   label = "PSA runs",        # label next to numeric input
                   value = 1000,              # initial value
                   min = 0,                   # minimum value allowed
                   max = 400),                # maximum value allowed

      sliderInput(inputId = "SI_n_age_init",  # id of input, used in server
                  label = "Initial Age",      # label next to numeric input
                  value = 25,                 # initial value
                  min = 10,                   # minimum value allowed
                  max = 80),                  # maximum value allowed


      actionButton(inputId = "run_model",     # id of action button, used in server
                   label   = "Run model")     # action button label (on button)

    ),  # close sidebarPanel

    mainPanel(                                # open main panel

      h3("Results Table"),                    # heading (results table)

      tableOutput(outputId = "SO_icer_table"),   # tableOutput id = icer_table, from server

      h3("Cost-effectiveness Plane"),         # heading (Cost effectiveness plane)

      plotOutput(outputId = "SO_CE_plane")       # plotOutput id = CE_plane, from server

    ) # close mainpanel

  ) # close sidebarlayout

) # close UI fluidpage
```

Server



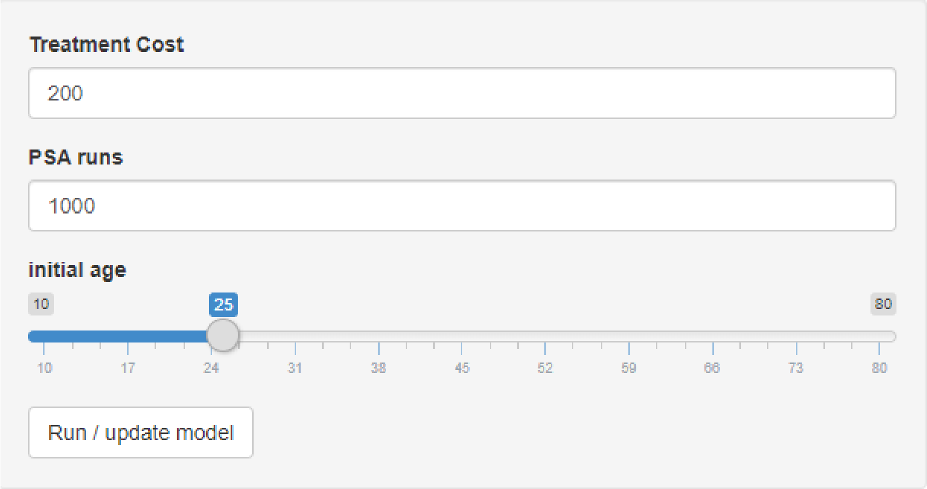
```
server <- function(input, output){   # server = function with two inputs

  observeEvent(input$run_model,       # when action button pressed ...
               ignoreNULL = F, {

                # Run model wrapper function with the Shiny inputs and store as data-frame
                df_model_res = f_wrapper(c_Trt = input$SI_c_Trt,
                                         n_age_init = input$SI_n_age_init,
                                         n_sim = input$SI_n_sim)


                #--- CREATE COST EFFECTIVENESS PLANE ---#
                output$SO_icer_table <- renderTable({ # this continuously updates table

                  df_res_table <- data.frame( # create dataframe

                    Option =  c("Treatment","No Treatment"),

                    QALYs  =  c(mean(df_model_res$QALY_Trt),mean(df_model_res$QALY_NoTrt)),

                    Costs  =  c(mean(df_model_res$Cost_Trt),mean(df_model_res$Cost_NoTrt)),

                    Inc.QALYs = c(mean(df_model_res$QALY_Trt) - mean(df_model_res$QALY_NoTrt),NA),

                    Inc.Costs = c(mean(df_model_res$Cost_Trt) - mean(df_model_res$Cost_NoTrt),NA),

                    ICER = c(mean(df_model_res$ICER),NA)
                  )

                  # round the dataframe to two digits so looks tidier
                  df_res_table[,2:6] <- round(df_res_table[,2:6],digits = 2)

                  #print the dataframe
                  df_res_table

                }) # table plot end.


                #---  CREATE COST EFFECTIVENESS PLANE ---#
                output$SO_CE_plane <- renderPlot({ # render plot repeatedly updates.

                  # calculate incremental costs and qalys from results dataframe
                  df_model_res$inc_C <- df_model_res$Cost_Trt - df_model_res$Cost_NoTrt
                  df_model_res$inc_Q <- df_model_res$QALY_Trt - df_model_res$QALY_NoTrt

                  # create cost effectiveness plane plot
                  plot(x = df_model_res$inc_Q, # x axis incremental QALYS
                       y = df_model_res$inc_C, # y axis incremental Costs
                       #label axes
                       xlab = "Incremental QALYs",
                       ylab = "Incremental Costs",

                       # set xlimits and ylimits for plot.
                       xlim = c(min(df_model_res$inc_Q,df_model_res$inc_Q*-1),
                                max(df_model_res$inc_Q,df_model_res$inc_Q*-1)),
                       ylim = c(min(df_model_res$inc_C,df_model_res$inc_C*-1),
                                max(df_model_res$inc_C,df_model_res$inc_C*-1)),
                       # include y and y axis lines.
                       abline(h = 0,v=0)
                  ) # plot end
                }) # renderplot end

              }) # Observe Event End


} # Server end
```

https://github.com/RobertASmith/healthecon_shiny/blob/master/App/app.R

Making the app run

But only when you want it to….

Making the run button



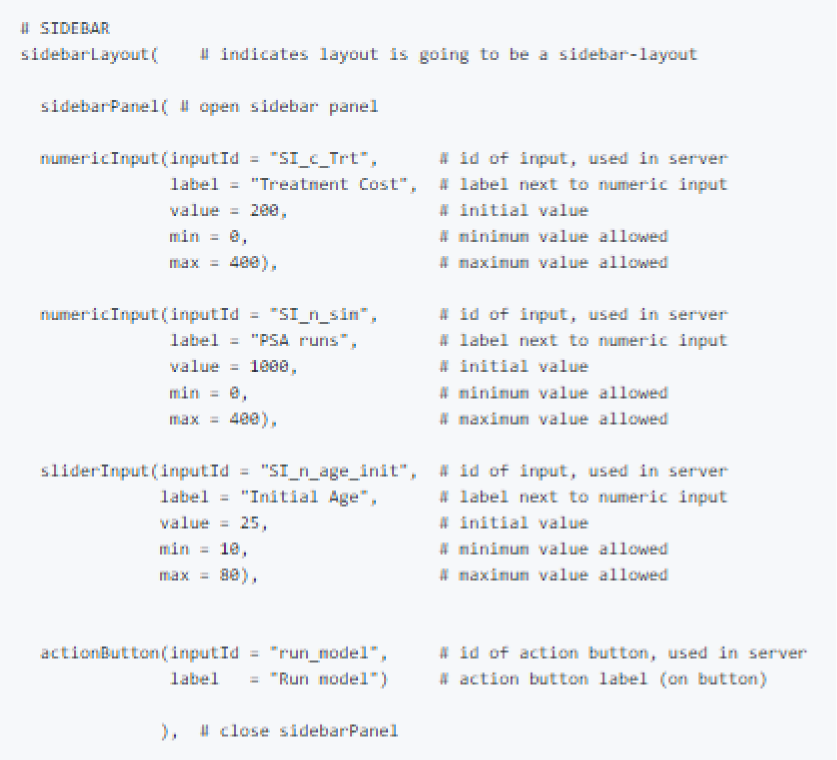
Treatment Cost

200

PSA runs

1000

initial age



```
# SIDEBAR
sidebarLayout(    # indicates layout is going to be a sidebar-layout

  sidebarPanel( # open sidebar panel

  numericInput(inputId = "SI_c_Trt",      # id of input, used in server
             label = "Treatment Cost",  # label next to numeric input
             value = 200,               # initial value
             min = 0,                   # minimum value allowed
             max = 400),                # maximum value allowed

  numericInput(inputId = "SI_n_sim",      # id of input, used in server
             label = "PSA runs",        # label next to numeric input
             value = 1000,              # initial value
             min = 0,                   # minimum value allowed
             max = 400),                # maximum value allowed

  sliderInput(inputId = "SI_n_age_init",  # id of input, used in server
            label = "Initial Age",      # label next to numeric input
            value = 25,                 # initial value
            min = 10,                   # minimum value allowed
            max = 80),                  # maximum value allowed


  actionButton(inputId = "run_model",     # id of action button, used in server
             label   = "Run model")     # action button label (on button)

            ),  # close sidebarPanel
```